Assigned Responsibility: An Architecture for Mixed Control Robot Teleoperation

Nicolas Small

Murdoch
UNIVERSITY

This thesis is presented for the degree of
Doctor of Philosophy of Murdoch University

October 2016

Abstract

Teleoperated robots are well suited to many tasks involving actions in hazardous environments due to their resilience and variety in sensory capabilities, size, tooling, and instrument-carrying capacity. Teleoperation can be achieved through a variety of control modes, ranging from direct human control of all systems to fully automated control. Some teleoperation systems use a range of these modes and encourage actively switching between them to suit the situation at hand. Most of these systems choose a reactive approach, relying on the controller (human or automated) to respond to changes in the task by switching appropriately.

The research presented in this thesis focuses on an unexplored gap in the spectrum of possible strategies for this mode switching. In situations where the environment is known and/or predictable, pre-planning these control changes could relieve robot operators of this additional task. Such a strategy provides a clear division of labour between the automation and the human operator(s) before the job even begins, allowing for individual responsibilities to be known ahead of time, thus limiting confusion and allowing breaks to be planned, for example.

This thesis proposes an assigned responsibility strategy, with an architecture supporting these pre-planned changes. A practical implementation of assigned responsibility is produced. This is evaluated through engineering tests and a usability study, demonstrating the viability of this approach as well as offering insight into its potential applications.

Declaration

I declare that this thesis is my own account of my research and contains as its main content work which has not previously been submitted for a degree at any tertiary education institution.

Nicolas Small

Acknowledgements

I would like to thank everyone without whom this work would not have been possible. First of all, a big thank you to my supervisors Graham Mann and Kevin Lee, who provided countless hours of useful advice and technical know-how, and whose office doors were always open. Secondly, thanks to everyone who volunteered their time to operate the robot for my experiments; your frustration was a source of valuable insight. Finally, I would like to thank the friends and family who provided moral support and much needed distractions when writing got to me.

Contents

List of Figures

List of Tables

Major Publications

Selected fully peer-reviewed conference, journal and workshop publications (most recent first).

Lee, K., Small, N., & Mann, G. (2015). Adaptive Planning for Distributed Systems using Goal Accomplishment Tracking. In *Proceedings of the 13th Australasian Symposium on Parallel and Distributed Computing (AusPDC 2015)*. Sydney, Australia.

Mann, G., Small, N., Lee, K., Clarke, J., & Sheh, R. (2015a, September). Field-Testing Astronaut Assistance Robots in Australian Outback [From the Field]. *IEEE Robotics Automation Magazine*, *22*(3), 188–191. doi:10.1109/MRA.2015.2452200

Mann, G. A. & Small, N. (2012). Opportunities for enhanced robot control along the adjustable autonomy scale. In *Human System Interactions (HSI), 2012 5th International Conference on* (pp. 35–42). IEEE.

Mann, G., Small, N., Lee, K., Clarke, J., & Sheh, R. (2015b). Standardized Field Testing of Assistant Robots in a Mars-Like Environment. In *Towards Autonomous Robotic Systems* (pp. 167–179). Springer International Publishing.

Small, N. J., Mann, G., & Lee, K. (2013, December). Goal Accomplishment Tracking for Automatic Supervision of Plan Execution. In *Proceedings of the 2013 Australasian Conference on Robotics and Automation*. Sydney, Australia.

Small, N. J., Mann, G., & Lee, K. (2015). Assigned Responsibility for Remote Robot Operation. In *Proceedings of the 16th Australasian User Interface Conference (AUIC 2015)*. Sydney, Australia.

Chapter 1

Introduction

1.1 Overview

Remotely operated robots[1] are well suited to many tasks involving dangerous actions in hazardous environments due to their resilience and variety in size, tool, and instrument carrying capability. The use of robots is not only justified by the need to prevent loss of human life but also often reduce the cost of operations as well as offer broad data gathering capabilities (Fong & Thorpe, 2001) (Rehnmark et al., 2005). These robots are often only partially autonomous [2], and thus require control by one or more human operators to function. As this control is remote, they are said to be teleoperated.

Teleoperation (from the Greek word-forming element "tele", meaning "far off", and the latin "opera" meaning "effort"), is defined in current academic literature as being the act of extending a person's ability to manipulate objects to a remote location, through the use of a control station (the master) linked to a manipulator (the slave) situated at the remote location (Hokayem & Spong, 2006).

Teleoperation requires the operator to make control decisions based on

[1] It is important to note that this text uses the word robot as a general term covering most machines/vehicles that are remotely operated, either automatically or by human operators.

[2] Autonomous and automated are used interchangeably throughout this thesis. Autonomous robots use automation instead of human control, and automated robots are considered to be autonomous from humans.

the information relayed by the robot about its own status as well as its surroundings. Often this information is limited in quality and quantity, placing operators under significant cognitive load during the control task. Some setups face the opposite problem, as sensor-heavy robots can provide the operator with a great amount of information, overloading the operator. This has lead to task specific displays being designed in order to make the reading of multiple sources of information more effective and efficient (Nielsen, Goodrich, & Ricks, 2007; Nielsen & Goodrich, 2006).

A solution to the problem of operator overload involves sharing the load between the human operator and the robot being controlled. The idea of shared responsibility (or adjustable autonomy, if the load on both operator and robot is changeable), was introduced in previous works which proposed multiple approaches to the problem (Goodrich & Schultz, 2007).

Most of these approaches focus on teams of multiple humans and robots working together on a more or less equal footing, whereas the research presented in this thesis will focus on simplifying teleoperation tasks, where a single operator is controlling one or more robots. The aim of shared responsibility systems is to combine the processing power of the robot and the ingenuity of the human operator, to address issues with both low-level teleoperation, where the operator is unable to neglect his charge, and full automation, which is currently still limited when performing any more than very specialised tasks (Kumar & Mason, 2011).

The remainder of this chapter is structured as follows. Section 1.2 establishes the need for remotely operated machines and highlights the challenges associated with that remote operation. Section 1.3 describes human driven teleoperation, while Section 1.4 covers the alternative: automated control. Section 1.6 then introduces the aims and contributions of the thesis while Section 1.7 presents the overall structure of the thesis.

1.2 Remote Operation of Robots

1.2.1 Overview

Remotely operated (or remote controlled) robots, are usually designed for and deployed in environments where their human counterparts may not operate. The need to keep humans away from varied environments has lead to the design of specialised robots, with four main design considerations being: i) the physical ability to perform the tasks they are replacing the human for, ii) the ability to record and understand the environment surrounding the machine, iii) the ability to communicate with a safe remote location, and iv) the ability to interpret commands received from that safe remote location.

The need to overcome these constraints has driven teleoperation and robotic automation research, two areas with considerable overlap. Teleoperation can be described as the process of directly controlling a remote vehicle or robot in order to make that vehicle or robot accomplish a task, effectively extending the reach of the operator. Robotic automation, especially when run directly on the vehicle, may go some way to eliminating the need for a constant remote communications link, by providing the machine with sufficient intelligence to allow it to make decisions regarding its course of action, again with the goal of accomplishing a task in mind.

1.2.2 Applications for Remotely Operated Robots

The search for solutions to problems with both teleoperation and robotic automation is encouraged by the large number of cases in which robots are deployed, such as explosive ordnance disposal, post-disaster rescue operations, military missions, space missions, deep sea exploration and maintenance (Scholtz, Theofanos, & Antonishek, 2006; Murphy, 2004; Fong & Thorpe, 2001). The decision to use robots is usually made when the scenario surrounding the accomplishment of a task falls inside one (or usually more) of the categories listed below.

- A task needs to be performed in an area that is too dangerous for a

human to safely operate in (e.g.: The inspection of the disaster sites following the Great Eastern Japan Earthquake using the KOHGA3 robot and the SARbot, Seamor, and AC-ROV underwater robots (Matsuno et al., 2014)).

- A task needs to be performed in an area inaccessible (or simply hard to reach) to humans (e.g.: The lunar surface exploration by the Chinese Yutu rover (Ip, Yan, Li, & Ouyang, 2014)).

- A task requires capabilities beyond those physically deliverable by a human being. (e.g.: The use of passive polarising stereoscopic displays to communicate minute depth cues to surgeons performing telesurgery (Smith et al., 2012))

1.2.3 Remote Operation Challenges

The remoteness of the robot being controlled introduces a number of challenges to the control scenario. In the context of interaction between an operator (human or automated) and the robot, four areas stand out as being of key importance for any solution to the remote control problem.

a. Limitations in Communications

Communications over long distances carry with them two main issues; i) limitations on bandwidth and ii) travel time related latency. The bandwidth issues most likely stem from the lack of quality networks over which to conduct the remote control operation. A lack of bandwidth impacts both the quality and quantity of information able to travel over the communications link at any one time. This is a critical limitation when there is a need to transmit bandwidth-hungry data such as video feeds. The problem of latency over a communications link is a consequence of three factors: transmission delay, processing delays, and queuing delay (Gettys & Nichols, 2012). The main impact of latency on the remote operation of robots is the introduction of a delay between the issuing of a command and the arrival of feedback from the robot. This delay can slow

down the task execution dramatically, as the operator has to wait for this feedback before initiating any new commands.

b. Acquisition of Situational Awareness

Knowing about the environment in which the robot is interacting is of crucial importance to making good control decisions. Unlike a more traditional control scenario, where the operator might be able to see the machine directly in its environmental context (e.g. a person driving a car), or where the environment is very well known (e.g. a factory floor), remote control tasks rely on feedback from sensors mounted on the robot to provide this situational awareness. The ability to record and communicate sufficient information about a machine's direct surrounds is key to the successful accomplishment of a remote control task. Without this information, next-action decisions made cannot be made with full confidence, and task-jeopardising errors might be made (collision with an unseen object for example).

c. Translating Operator Intent to Machine Commands

An operator, human or automated, will need to make decisions on what the robot should do, and be able to communicate these decisions for remote execution. Translating this intent into physically executable commands is a problem in itself. This translation needs to be done at some level between the operator and the robot to ensure that there is no disparity between what the operator wants to do and what the robot actually does. In some implementations, the operator might need to have a clear understanding of how a particular robot performs a task, in order to be able to do this translation directly, providing relatively low-level commands. This is in contrast to other setups where only high level commands are expected from the operator, and an abstraction layer between the operator and robot is charged with this translation task.

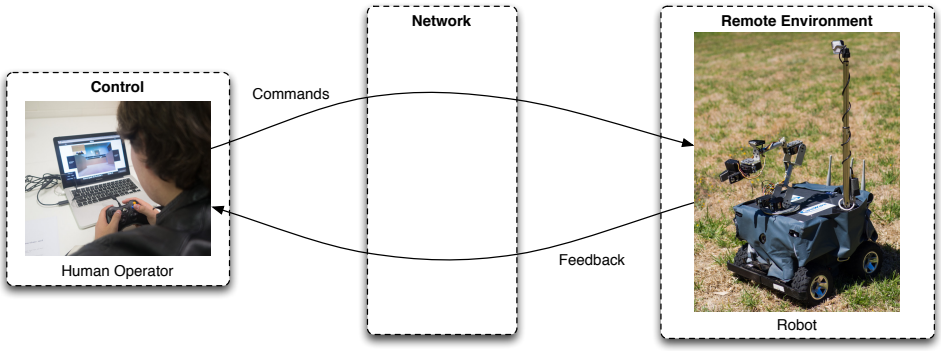


Figure 1.1: The basic Teleoperation loop.

d. Physical Capabilities

The physical capabilities of a robot can limit the range of tasks in which it can replace direct human involvement. This aspect covers many features of the vehicle, such as the presence and type of manipulators, battery life etc. While being overall a very important requirement which has a significant impact on robot design, the main consideration of this thesis is on the interaction between the operator and the robot. As such, the interest in the makeup of the robot stems from the impact the presence or absence of features will have on how the operator controls it, rather than the amount of force a manipulator can apply, for example.

1.3 Human Teleoperation of Robots

1.3.1 Overview

Teleoperation is one of the solutions presented here as a way of controlling robots at a distance, it is also the oldest and most widely used paradigm for this type of control. The human operator sends out commands to the robot using a control station that may take many forms, and receives information back from the robot's mounted sensors (as depicted in Figure 1.1).

1.3.2 Real-World Examples of Use

While automated control techniques such as those described in Section 1.4 are gaining in popularity, issues with reliability and distrust in non-human controlled systems mean that the relatively old teleoperation paradigm is still the preferred option for robot control. The areas of use presented in Section 1.2.2 mainly make use of direct teleoperation except for a few examples. The following are examples of applications of teleoperation, showcasing the wide range of technologies and designs used in these tasks.

a. Explosive Ordnance Disposal

One of the early motivations for the development of teleoperated vehicles, explosive ordnance disposal (EOD) presents a classic use-case for the deployment of robots (Figure 1.2 shows an EOD Robot). To prevent the loss of life of bomb technicians, a way of extending an operator's reach to a bomb's proximity while keeping them at a safe distance is required. Tasks associated with EOD include navigation, manipulation, and the delivery of payloads to disrupt explosives. All of these tasks are performed by an operator using a portable teleoperation station (Carey, Kurz, Matte, Perrault, & Padir, 2012).

b. Telesurgery

Telesurgery seeks to make use of teleoperation techniques to improve surgical procedures in two ways: i) by allowing a trained practitioner to operate on a remote patient, and ii) by allowing surgeons to perform keyhole surgery, bypassing the need for large intrusions in the bodies of patients (Wall & Marescaux, 2013). Telesurgery platforms, like the Da Vinci Surgical System, make use of innovative control solutions to allow surgeons to control complex robotic arms (Palep, 2009).

c. Robonaut 1 and 2

Robonaut is a humanoid robot developed by NASA in collaboration with industrial partners (Diftler, Culbert, Ambrose, Platt Jr, & Bluethmann,

Figure 1.2: Explosive Ordnance Disposal (EOD) robot used by the Western Australian police force.

2003). The goal of this robot is to assist astronauts performing dangerous maintenance missions in space, and since 2011 has been present on the International Space Station (Diftler et al., 2012). Robonaut 2 can be operated using a GUI that lets the operator build up a sequence of actions, or more directly through tracking the motions of the operator through wearable gear, mapping the robot's motions on those recorded from the operator. This configuration is intended to make the operator feel like they are at the remote location, reducing the disconnect between operator and machine (Diftler et al., 2012).

1.3.3 Teleoperation and the Remote Operation Challenges

Teleoperation interfaces have to overcome the general challenges of remote operation presented in Section 1.2.3 in order to provide an operator with the level of control required to properly perform tasks with a robot. This section discusses how these challenges apply to teleoperation and the solutions used to overcome some of these problems.

a. Limitations in Communications

Bandwidth Issues: In order to indulge the human propensity to rely on visual stimuli, transmitting video feedback from the ROV to the operator is essential. Video feeds tend to require large amounts of bandwidth (especially when compared to other forms of telemetry), and as such, limitations to bandwidth impact the quality of the video that may be practically returned to an operator. Lack of quality video can severely impair human operators, and by extension affect the quality of work performed by the robot (Chen, Haas, & Barnes, 2007). As most teleoperation interfaces rely heavily on video as a form of feedback, a certain amount of bandwidth has to be assured before teleoperated work can be done. Chen et al. (2007) surveyed research on this topic and reported that video should be transmitted at no less than 10 frames per second. Below that figure, control quality is significantly affected.

Impact of Latency: An operator needs to be able to observe the result of their actions to gauge how successful they were. This means that human operators tend to wait for sensor feedback to arrive to them before initiating any further actions. This move-and-wait strategy (Hokayem & Spong, 2006) is effective but time consuming, making the completion time for any given teleoperated task a function of operator reaction time, time delay and number of actions to be performed to ensure successful accomplishment of that task. Complex tasks in high latency situations can become extremely time consuming and therefore frustrating for the operator. Some interfaces attempt to bypass this issue by including a predictive display that shows how the robot should be positioned and/or configured after a set of commands, allowing the operator to correct mistakes without having to wait for the real feedback (Bejczy, Kim, & Venema, 1990). Chen et al. (2007)'s survey of user experiments on latency in teleoperation, reported minimum acceptable latencies (above which control would be degraded) ranging from 170ms to 1s. This variation depends on the task undertaken, with driving tasks and complex manipulation tasks requiring low latencies, while tasks such as free flight

simulation are more forgiving.

b. Acquisition of Situational Awareness

Lack of Situational Awareness: A lack of situational awareness can introduce many unwanted factors to robot control. The first of these is significant time delays, as the operator is forced to carefully perform operations while possibly having to scan the environment with insufficient sensors (Murphy & Burke, 2005). This also requires the operator to memorise what has been seen and is currently out of shot. Both this memory requirement and the added threat of collision with the environment add significant mental strain on the human operator (Scholtz, Young, Drury, & Yanco, 2004). Recent developments in 3D imaging sensors are helping solve this problem, as it is now relatively easy to mount cheap 3D sensors capable of building a 3D map of a location to robots. This map can then be used by the operator to get a sense of the environment surrounding the robot (Liu & Nejat, 2013).

Display of Sensor Readings and Information Overload: Presenting the operator with the sensor readings from the robot brings its own set of challenges. Displays have to provide all the information needed by the operator to make informed decisions without overloading the operator's mind with that information. This balancing act between too much and not enough information can be eased through good interface design, by overlaying related information to ease understanding for example (Baker, Casey, Keyes, & Yanco, 2004; Fong, Thorpe, & Baur, 2001; Drury, Scholtz, & Yanco, 2006).

c. Translating Operator Intent to Machine Commands

Control Paradigms: Different solutions to the problem of translating the operator's intent to some form of machine readable commands have been offered in the literature (Fong & Thorpe, 2001). The most common of these is the "joysticks and screen(s)" style interface which is the standard setup for most teleoperation applications. Joysticks encode the operator's

intent while screens display feedback from the remotely operated machine. Other paradigms include telepresence, where the operator is made to feel as if they are at the remote location through the use of of immersive technologies. These include head-mounted displays and natural interfaces like cyber-gloves (Goza, Ambrose, Diftler, & Spain, 2004), and natural language interfaces, where the machine is controlled through voice and gesture-based commands (Perzanowski, Schultz, Adams, & Marsh, 1999).

Teleoperation Aids: To improve usability, reduce load on the operator, and/or save time, some teleoperation interfaces make use of automated aids. These aids are usually relatively simple in nature, and tend to abstract some of the more complex control schemes to make teleoperation easier. Examples of these aids include inverse kinematic control, that allows operators to set positions for the end effector on robotic arms, letting the automated component calculate the angle each of the joints on these arms need to be at, rather than having the operator set those joint angles individually (Das, Sheridan, & Slotine, 1989). Another example is the stabilisation algorithms for multirotor drones, which keep those drones stable during flight, allowing operators to focus on directing the drones rather than have to both direct and stabilise them (Tayebi & McGilvray, 2004).

This introduction of automated solutions in more traditional control schemes has already happened in other domains, such as with airplane control. In cases of partial automation, it is imperative to ensure the human operator is very clear on which parts of the system are automated, because conflicts between automation and human operators have already caused dramatic accidents (Parasuraman & Riley, 1997).

1.3.4 A Human in the Loop

The presence of a human in the control loop in teleoperation setups brings in more advantages than a simple reduction in programming complexity. A human operator carries with them a large amount of experience on how the world works, how objects can be interacted with, and the ability to produce creative solutions to previously unencountered problems. These abilities lend the teleoperation paradigm desirable characteristics such as robustness

and adaptability in the face of changing situations, and the potential to operate in unexplored environments.

Another benefit to having a human in the control loop is the opportunity for the robot to learn skilled action from a human expert, delivered directly to the machine in real contexts (Mann & Small, 2012). Learning by Demonstration (LbD) is a machine learning approach involving the extraction of a robot control policy from a set of demonstrations of the target skill, recorded from the sensors and motor outputs (Argall, Chernova, Veloso, & Browning, 2009).

1.4 Automatic Control of Robots

1.4.1 Overview

This section describes the use of automatic methods to control machines remotely and locally. Automation is usually performed on board the robot. As such, its deployments in remote control scenarios tend to face problems more from a particular system's understanding (or lack thereof) of the environment rather than the difficulties associated with communications over long distances. While not usually considered to be teleoperated, most automated vehicles operating in remote areas are accomplishing tasks set to them by a human supervisor, and are thus extending the reach of that supervisor, fitting the broad definition of teleoperation presented in 1.1.

1.4.2 Real-World Examples of Use

Automated robots are preferred to their human controlled counterparts in some applications, either due to increased efficiency, to the risks, costs or complications of using a human workforce, or to the difficulties tied with the process of remote operation. The examples below describe several applications for automated robot control systems, highlighting the reasons automated systems were chosen over their human driven counterparts.

a. Factory Automation

The predictable and well described nature of assembly lines have proven to be an ideal stage for automation. Very simple automation techniques can be employed in these scenarios, as the use for robots is often limited to the performance of pre-programmed repeating tasks (Brogårdh, 2007). The automotive industry has made successful use of automation in its factories, where robots can be found performing tasks such as car assembly and painting.

b. Mine Automation

A more recent commercial use of automation has come from the mining sector, with mining companies beginning to automate some of the heavy machinery used on mine sites. With an environment less structured than a factory floor, this automation tends to be more complex, relying on input from sensors such as GPS and vision-based sensors. For example, automated ore carrying trucks follow pre-planned paths between destinations, as well as making use of LIDAR (Light Detection and Ranging) sensors to detect and avoid obstacles in their paths (Brown, 2012).

c. Space Exploration Automation

Space exploration is an area with many uses for automation. With robots and probes as far as interplanetary distances away, direct remote control becomes impossible due to the extreme latencies involved, as well as limited windows of control time due to occlusions by planets etc. These scenarios require the use of automation to keep the remote machine safe during these periods of no contact. In the case of Mars exploration for example, robots on the ground have to deal with mostly unknown environments, as well as the near-impossibility of repairs (software updates being possible). These machines tend to be given high level commands that they then execute very carefully (Bajracharya, Maimone, & Helmick, 2008).

1.4.3 Automation and the Remote Operation Challenges

Automation has to overcome three of the challenges presented in Section 1.2.3 in order to properly perform tasks with robots. The fourth, regarding physical capabilities, being out of the control of an automated system.

a. Limitations in Communications

In an automated context, communications issues only really arise when the automation is performed from a location other than that of the vehicle. In that case, the issues are similar to those encountered in teleoperation scenarios, with video streams (if required) demanding high bandwidth, and the automation needing to be able to handle any time delays. When the automation is performed on board the vehicle however, the only information passed through a communications link tends to be relatively low bandwidth high-level commands (such as "Go To Waypoint A"). Latency is also of little concern for an automated controller running directly on the controlled machine, since the delays are likely very small.

b. Acquisition of Situational Awareness

To make useful decisions during operation, automated control systems need to either be able to collect information about the environment the robot is in and extract information about potential interactions with that environment, or already be in possession of an accurate description of that environment. Outside of carefully constructed and static environments, like factory floors, it is difficult to be in possession of all of the information required for a robot to operate autonomously. In unexplored or changing environments, it is imperative for the automatic controller to be able to build this understanding of its surrounds dynamically. With the accelerating improvements in sensor technologies (such as 3D scanning), this task is becoming easier, allowing robots to dynamically map their environments using simultaneous localisation and mapping (SLAM) techniques (Durrant-Whyte & Bailey, 2006). Beyond avoiding obstacles and navigating inside a location, automated robots need to be able to interact with objects in that environment, requiring an understanding of

what objects are present and how the robot can affect these (Collet, Berenson, Srinivasa, & Ferguson, 2009).

c. Translating Operator Intent to Machine Commands

Not unlike the translation problems encountered in teleoperation setups, the decision engine of an automated controller must be able to communicate its desired course of action to the robot's hardware controllers. This translation task can again be done at multiple levels, either by the automated operator or by an abstraction layer between this controller and the robot. Automation has the advantage here over teleoperation however, as there is no need for a human-machine interface to convert human intent to digital information. This removes one barrier between operator and robot, eliminating errors relating to usage of the control interface.

1.4.4 Automation in the Loop

Factors such as stress, fatigue, and frustration simply do not factor in the performance of automated systems. While these systems may lack the creativity in problem solving, and vast experience in interacting with the world that their human counterparts possess, automated controllers are well suited to performing long, repetitive, and potentially boring tasks that humans find hard to execute competently. Due to their repetitive and systematic nature, these tasks also tend to be the easiest to automate, making them prime targets for automatically controlled robots.

One of the main barriers to the deployment of automatically controlled robots is linked to how they are perceived by the people in charge of scenarios where they could be deployed. There is an inherent distrust of fully automated systems in circles outside of academia, due most likely to the lack of knowledge of the actual competencies and reliability of such systems (Lee & See, 2004).

1.5 Opportunities for Improving the Remote Operation of Vehicles

It is apparent that the remote operation of machines is an area with room for improvement. Both solutions to the problem, teleoperation, and automation, are lacking in several key aspects. Teleoperation requires the presence of human operators generally prone to stress, fatigue, and a dislike of simple repetitive tasks, while automated control systems lack the ability to operate unassisted in complex or unknown environments. Ideally, these automated systems will improve to the point where they can operate at full efficiency in these environments. This makes automatic control of ROVs the more desirable end goal, albeit one currently out of reach for many deployment scenarios.

An interim solution exits however: the merging of automation and teleoperated control. This trend has already begun with the introduction of teleoperation aids as discussed in Section 1.3.3, and is easily justified when the capabilities and failures of both automated and teleoperated control are compared. The lack of creativity and environmental understanding displayed by automated control systems is easily compensated for by the inclusion of a human operator in the loop, while the repetitive tasks which stress and tire human operators are usually relatively easily automatable. This approach can also be used to help progress towards the full automation ideal, as the inclusion of a human in the control loop can be exploited to gather some valuable training data for automated systems (See Section 1.3.4).

This sharing of tasks based on complimentary capabilities requires an understanding of both these capabilities and the tasks to be performed, and will need a flexible framework to be successfully executed.

1.6 Aims and Contributions

While full automatic control of robots is likely the ideal for many tasks, it is reasonable to assert that current automation techniques are still too limited when tackling complex real-world scenarios. The alternative,

human teleoperation of those same robots, while being the current commercial practice, is less than ideal as it places human operators under considerable stress and frustration. Current experimental platforms are merging the two, with semi-automated robots shouldering some of the burden, but still face limitations with automation, forcing the human operator to take over in complex situations.

The main aim of the research presented in this thesis is to propose a novel theory of teleoperation called Assigned Responsibility. By breaking down tasks into smaller sub-components, and explicitly allocating these components to the human operator and/or the automation (guided by their respective skills and capabilities), this Assigned Responsibility seeks to further the gains made by semi-automated teleoperation systems while providing the support for further automation of a task.

This research will seek to answer the following questions:

- Is Assigned Responsibility an effective, efficient, and satisfying model for the teleoperation of robots?
- Is the explicit allocation of tasks, between robot and operator, an effective way of reducing the workload placed on the operator by traditional teleoperation systems?
- Can Assigned Responsibility expand the capabilities of a robot beyond those currently automatable?

Contributions of this research include:

- The idea of and an architecture for Assigned Responsibility
- The idea of and an architecture for Goal Accomplishment Tracking
- Experimental evidence supporting the theory of Assigned Responsibility.

1.7 Structure of the Thesis

The thesis is structured as follows.

Chapter 2 reviews the current state of teleoperation interfaces. A study of the issues facing human operators and of the technologies in place to alleviate the process of teleoperation is provided. The use of partial automation strategies as an aid to teleoperation is also assessed. Finally, this chapter offers a comparative analysis of teleoperation systems across a wide range of applications.

Chapter 3 provides background on adjustable autonomy systems. A discussion on the basics of these systems and how they can expand the capabilities of more traditional teleoperation is provided. An analysis of five implementations of adjustable autonomy systems is done, focusing on the levels of autonomy they employ, as well as the strategies these systems use to decide when and how to change the level in use.

Chapter 4 introduces the theory of assigned responsibility for robot teleoperation. The need for such an approach is argued based on the findings of Chapters 2 and 3. An assigned responsibility-suitable automation scale is presented, and an assigned responsibility architecture is introduced.

Chapter 5 presents details of an implementation of the assigned responsibility architecture, as well as details of the robotic platform, and automation and teleoperation components. These implementations are validated according to the requirements presented in earlier chapters.

Chapter 6 evaluates the system end-to-end through the testing of the implementation. This experimental approach relies on user testing of the system. The experiments conducted are described and their results analysed.

Chapter 7 concludes the thesis, reiterating the main points of the research and offering potential avenues for future research in the area.

Chapter 2

Teleoperation

2.1 Overview

Teleoperation, the process of remotely executing actions through the use of a robot, is of great use in many domains where dangerous work needs to be performed (Sheridan, 1992). The ability to remove human workers from direct contact with dangerous environments through the use of technology is very desirable and motivates the continued research into teleoperation.

Although the advantages of removing workers from danger are clear, interaction with environments by proxy presents its own set of issues. Operator decisions on how to act are now based on the limited information returned by the machine at the operating end of the system and actions are limited by the capabilities of the interface they must use to communicate their commands. The success of teleoperation systems thus relies on balancing the safety gained from removing human workers from dangerous environments with enabling their work to still be performed adequately (Hainsworth, 2001).

This chapter presents the current state of teleoperation through the analysis of key representative systems in different domains. These systems are analysed through their operational contexts, their integration (or not) of automation, and their consideration for user experience.

The remainder of this chapter is as follows, Section 2.2 describes the fundamentals of teleoperation, covering the basic model of teleoperation,

as well as breaking that model down into its components to describe those in detail. Real world uses of teleoperation are then described to provide context and motivation for current teleoperation research in Section 2.3. Human-robot interfaces and their application to the wide variety of teleoperation techniques and paradigms are investigated in Section 2.4 with specific implementations of teleoperation analysed in Section 2.5. Finally, the difficulties faced by teleoperation systems are discussed in Section 2.6.

2.2 Teleoperation

2.2.1 Overview

In teleoperation, the operator and machine are separated by a barrier of some sort (Fong & Thorpe, 2001). The nature of this separation can vary; both walls and physical distances (short and long) place constraints on the ability of an operator to control a machine. Overcoming the problems arising from this separation of operator and machine is the essence of teleoperation research (Glass & Briggs, 2003). These include difficulty in communications, inadequate controls, and insufficient feedback and are discussed in detail further in this chapter.

There is some debate as to how much automation is allowed in the process before teleoperation can no longer be properly called teleoperation, with some authors only allowing direct human control while others also include systems only requiring high level commands from the operator (Fong & Thorpe, 2001), (Sheridan, 1992). This chapter considers teleoperation to be any system that allows a human operator to extends their reach beyond their immediate surroundings. This ranges from direct control teleoperation, to autonomous machines programmed to perform tasks at a distance.

The remainder of this section presents the fundamentals of teleoperation, describing the interactions between the different building blocks that make up a typical teleoperation system. The technical challenges faced by these systems are then elaborated to provide context for the rest of the chapter.

2.2.2 The Basic Elements of Teleoperation

The most basic model of teleoperation requires three elements: an operator, a remote machine, and a communications system (including an operator interface) to link the both of them (Sheridan, 1992). Figure 2.1 illustrates a model of basic direct control teleoperation system as described by Sheridan (1992). The "direct control" term characterises the type of control exercised by the operator: unmediated control over all of the robot's actuators. This is in contrast to other control strategies available to teleoperation systems as described later in Section 2.4 where some kind of computer system intermediates. While not the only paradigm, direct control is the most stripped-down version of teleoperation and provides a good starting model for teleoperation as a whole, that other strategies add to rather than replace.

This model consists of an operator and a robot, linked by a control and sensor loop. The operator inputs commands at a control station, the robot executes these commands using its actuator(s), the robot's sensors record this, and the operator can inspect this sensor data on a display. The operator's station and robot are connected by a communications network that must be able to transport the information required by the control loop. Most research in teleoperation focuses on one or more of these three areas: the operator's station, communications, and the robot.

2.2.3 Technical Challenges in Teleoperation

Teleoperation systems face a number of technical challenges during both their development and their deployment. These challenges are varied in nature due to the inherent complexity and multi-disciplined nature of teleoperation. These challenges can be meaningfully grouped by the element of the system they affect the most, as follows.

a. Operator-Related Challenges

The operator is the initiator of the tasks the robot performs. The role of the operator is to decide on a course of action and communicate that course of action to the robot. These decisions made by the operator rely

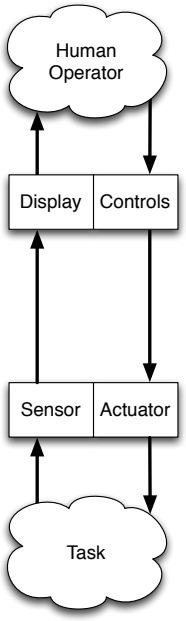


Figure 2.1: Elements of a direct control teleoperation system (Adapted from Sheridan, 1992).

on the feedback provided by the robot. This places two requirements on a teleoperation system: the system must provide the operator with a way of commanding the robot, and the system must be able to present the feedback from the robot's sensors to the operator. These provisions are normally combined in an operator's control station.

Olsen and Goodrich (2003) identify six metrics for evaluating a human-robot team. While these were designed to consider all interactions between a human master and robot servant, regardless of control paradigm, they are applicable to teleoperation as a subset of these interactions. The following presents these six metrics from Olsen and Goodrich (2003) with a short explanation.

- Task effectiveness: How effective the human-robot team is at accomplishing tasks.
- Neglect tolerance: How long the robot can continue accomplishing tasks without human input.
- Robot attention demand: The amount of attention from the operator

required by the robot.

- Free time: Any time spent by the operator not attending to the robot.
- Fan out: How many robots a single user can effectively guide through a task at once.
- Interaction effort: A measure of the effort required by the operator to control the robot. This can be physical but tends to be mostly cognitive.

Apart from task effectiveness, traditional teleoperation interfaces tend to score poorly with these metrics, as the control modality is very restrictive. The constant need for control by the human operator (due to the lack of autonomy in the robot) means there is a low tolerance for neglect, high robot attention demand which leads to low free time (Olsen & Goodrich, 2003). Generally speaking, teleoperation tends to be a one-to-one affair, which means a fan out of one (this is of course not true of cases where a robot is controlled by more than one operator and vice-versa, but describes most teleoperation use cases).

With high robot attention demand and low free time allowed by the paradigm, interaction effort becomes the main area for improvement in traditional teleoperation interfaces. This means facilitating the commanding of the robot, and making the taking in of robot feedback as easy as possible. Drury, Scholtz, and Yanco (2004) introduced the concept of *HRI (Human-Robot Interaction) Awareness*, which can be summarised as the knowledge each participant in a system has of the other participants. In the case of teleoperation, this means the operator's understanding of the state of the robot, and the robot's knowledge of the operator's commands. A desirable teleoperation interface should strive to provide high HRI awareness for a low level of interaction effort.

In removing the operator from the work site, teleoperation interfaces must act as a proxy for the senses of operators to allow them to increase their awareness of the robot and its environmental context. Sight is the sense that is leveraged the most in teleoperation interfaces, followed by touch (Hokayem & Spong, 2006). One or more video feeds from the robot tend

to make up the majority of the information used by the operator. While video provides some visual information, it tends to provide the operator with a rather poor replacement for human sight. Video streams used in teleoperation systems have been found to suffer from or cause many issues, such as insufficient or overly wide fields of view, lack of peripheral vision, multiple cameras dividing the attention of the operator, motion sickness, low image quality and frame rate, and lack of depth perception (Chen et al., 2007; Mann, Small, Lee, Clarke, & Sheh, 2015).

b. Communications Challenges

Due to the distributed nature of teleoperation, a reliable communications network is required to allow operator and robot to communicate during the task. Deployment scenarios make use of different vectors (wired or not, analogue or digital), but all seek to provide a stable, low-latency link with enough bandwidth to carry the necessary data.

Latency is a big problem for teleoperation systems, as any delays in communications affect how fast an operator's commands are sent, and how soon they see the feedback from their actions. This in turn impacts the time in which they can react to events occurring on the robot's side, possibly causing unwanted incidents such as collisions. Latency also takes a toll on the operator mentally, as adjusting to the delay requires additional concentration from the operator (Sheridan, 1992). In high latency situations, operators tend towards a move-and-wait solution, where they command the robot, and wait for the feedback to reach them before initiating any other commands (Hokayem & Spong, 2006). In extremely high latency situations, such as in the case of rovers operating on Mars, latency is in the order of minutes, rendering direct teleoperation approaches useless (Biesiadecki, Leger, & Maimone, 2007). This has lead to alternate models of teleoperation being developed to allow the interplanetary control of these rovers (See Section 2.3).

The bandwidth of a particular communications link limits the throughput of that link. This has a significant impact on teleoperation systems, as a low bandwidth link can limit the amount of sensor

information available to the operator, thus slowing decision making and action monitoring. It also affects the power requirements of the robot and the control station, as higher bandwidth tends to require more power.

Latency and bandwidth issues should thus have a significant impact on the design and implementation of a teleoperation system. The direct teleoperation model described above is best suited to use cases where low latency and high bandwidth communications are sustainable, as it usually relies on operator reactions (time-sensitive) and video feedback (bandwidth-intensive) (Fong & Thorpe, 2001).

c. Challenges in Robot Design

As the physical presence at the remote location, the robot needs to be capable of performing two tasks: i) affecting its environment as commanded, and ii) recording that environment for monitoring by the operator. The first task is an engineering problem, ensuring the robot is physically capable of performing the tasks required of it. The second task, while also an engineering problem (how to design and mount sensors on the robot) is a matter of human factors. Which sensors to select, and where they should be positioned on the robot, as well as how to use them to inform the operator at their station are factors that can affect the operator's ability to accurately command the robot (Chen et al., 2007).

2.2.4 Summary

The aim of teleoperation is allowing a person to interact with an environment without being present in that environment. It is apparent that the challenges presented above mostly impact the quality of the integration of the operator into the teleoperation system, which is logical when this purpose is considered.

The quality of this integration can be affected by two factors. Firstly, by the interface used by the operator, which directly affects the ability of this operator to understand the state of the robot, and issue commands to that robot. Secondly, this integration is also potentially jeopardised by the quality of the communications link between operator and robot, and the

capabilities of the robot. A low quality integration can cause the operator to be stressed, fatigued, and/or frustrated beyond necessary levels, all of which can lead to mistakes, and at the very least to reduce task efficiency (Chen et al., 2007).

2.3 Use Cases for Teleoperation

2.3.1 Overview

While a useful starting point for the design of a teleoperation system, the basic model presented in Figure 2.1 is not always directly applicable to a particular use case. Environmental factors and/or other constraints can mean that modifications to that model need to be made before teleoperation in that environment is possible or effective. This section details three deployment scenarios, all of which are real-world examples. These scenarios are analysed, and the constraints they place on teleoperation systems are outlined. Finally, for each scenario, a list of requirements for a teleoperation system acting in those environments is produced.

2.3.2 Urban Search and Rescue

Urban Search and Rescue (USAR) missions are mainly concerned with the search for victims in post-disaster urban environments. These missions often require robots to be sent to hard to reach confined spaces such as voids below rubble (Murphy, 2004).

a. Role of Teleoperation

Unlike most teleoperation use cases, USAR scenarios require teleoperated robots to be deployed not to remove humans from dangerous situations, as these robots will most likely be working alongside rescue workers in the field, but mainly to search spaces these rescue workers cannot reach, or deemed too dangerous (Figure 2.2 shows the KOHGA3 robot inspecting the structural state of a damaged gymnasium that was barred from human access). The role of operators and robots is to search for any signs of victims trapped

Figure 2.2: The KOHGA3 robot in a damaged gymnasium after the 2011 Great Eastern Japan Earthquake (Matsuno et al., 2014)

in the rubble, and alert rescue workers to what they find (Liu & Nejat, 2013). The robot might also be expected to act as an intermediary between a conscious victim and the rescue team, relaying communications from both using a microphone and speakers (Murphy, 2004).

b. Environmental Challenges

The operating condition of USAR missions place restrictions on both the size of the robot, and usable communication technologies. With size restrictions come other limitations, as small robots lack the carrying capabilities of larger robots, preventing them from carrying large batteries and limiting the size and number of sensors they can take. Because of this, USAR robots tend to be tethered, which allows them to be lowered down and lifted up from vertical shafts by the rescue workers present at the site (Murphy, 2004). The tether also allows USAR robots to bypass the communications problems caused by this environment, as wireless communications can be unreliable in urban areas due to intervening obstacles that interfere with the radio link.

Beyond robot capabilities, USAR missions place significant strain on operators, as they are required to perceive, understand, and navigate complex environments while searching for survivors (Liu & Nejat, 2013). This task has been found to be difficult enough that researchers have recommended it be undertaken by two operators at once: one navigating, the other reviewing the sensor data from the robot (Murphy, 2004).

c. Solutions

USAR missions require specific capabilities from their human-robot teams. The robots used in USAR need to be capable of performing in complicated unstructured environments. This might require them to be of an appropriate size, have the ability to be retrieved using a lifeline/tether, and have the ability to maintain communications in unknown and highly disrupted environments. The systems are equipped to allow the identification of trapped victims through the use of appropriate sensors (video, lighting, and IR cameras), and the ability to have that sensor data be reviewed by multiple people (the operator and a rescue worker for example). Finally, USAR teleoperation systems may need to allow two-way communications between operator/rescue workers, and victims or other rescue workers.

2.3.3 Robot-Assisted Keyhole Surgery

Robot-assisted surgery consists in using a teleoperated robot to perform surgical operations. Unlike usual applications of teleoperation, the surgeon is in the same room as the robot and patient, but still operates the robot through video feedback, as the minimally invasive nature of the surgery means direct observation is impossible (Palep, 2009). Figure 2.3 shows the surgeon and patient sides of the da Vinci telesurgery system.

(a) Surgeon side. **(b)** Patient side.

Figure 2.3: Photographs of the Da Vinci telesurgery system (Sung & Gill, 2001).

a. Role of Teleoperation

An evolution of hands-on laparoscopic (keyhole) surgery, robot-assisted surgery removes the surgeon from direct contact with the patient to allow them to operate through the use of a surgical robot. It is important to note that traditional laparoscopic surgery is not so dissimilar to teleoperation in itself, as the surgeon operates using laparoscopic instruments that extend their reach into the body of the patient while acting based on feedback from a camera operated by an assistant (Marohn & Hanly, 2004). The motivation behind replacing the laparoscopic instruments with a surgical robot is to provide the surgeon with better control over the manipulators through increased degrees of freedom, direct control over the camera (rather than having to communicate intentions to an assistant), and enabling more surgeons to be able to perform these complex surgeries (Marohn & Hanly, 2004).

b. Environmental Challenges

The co-location of the surgeon and robot removes most of the communication issues faced by more usual teleoperation systems (i.e.: dealing with long distances or physical obstacles between robot and operator), which means the issues faced by these surgery systems lie at the manipulator end, and the operator interface (Camarillo, Krummel, & Salisbury, 2004).

Being located inside a patient during the operation, the robot's manipulators are likely to cause grave damage if misused or used carelessly. Complete control over the manipulators is therefore of major importance in a system such as this, as well as proper awareness of the manipulators' location and surroundings. In addition to this, the surgeon may be operating in very small areas, requiring highly precise motions to be performed (Camarillo et al., 2004).

c. Solutions

Surgical robots used in laparoscopic surgery provide surgeons with capabilities beyond what they had when performing traditional laparoscopic surgery. To avoid manipulation errors, modern surgical

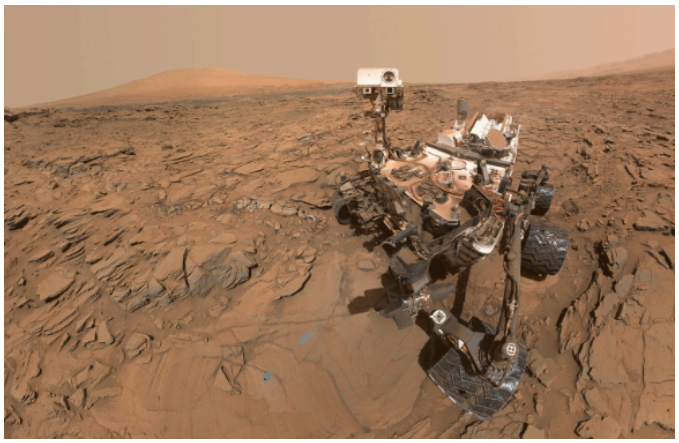


Figure 2.4: MSL Curiosity's self-portrait at the Okoruso Drill Hole on Mars (mars.nasa.gov, n.d.)

robots provide surgeons with 3D vision, to help with depth perception issues (Smith et al., 2012), tremor abolition systems to mitigate the effects of inadvertent hand shaking, and motion scaling to allow surgeons finer control over the motions of the manipulators, scaling large motions down to the fine motions required by the surgery (Palep, 2009).

2.3.4 Mars Exploration

Attempting the teleoperation of a robot at interplanetary distances is an unsurprisingly difficult task. With varying amount of delays in communications (anywhere between 8 and 42 minutes) (Biesiadecki et al., 2007) and limited windows for those communications to take place in, operating robots on the surface of Mars from Earth requires a different approach than direct control teleoperation.

a. Role of Teleoperation

In the absence of human astronauts on the surface of Mars, its exploration has been done through the rovers Sojourner, Spirit, Opportunity, and Curiosity (Pictured in Figure 2.4). These missions have so far been focused on performing geological surveys of the surface and searching for water and signs of life either past or present. To do this, the rovers are/were high level commanded to traverse the Martian landscape, photograph areas of

interest and perform sample analysis of the surface (Washington et al., 1999; Biesiadecki et al., 2007; Grotzinger et al., 2012).

b. Environmental Challenges

The primary obstacle to teleoperation in this deployment scenario is the distance between operator(s) and the robot. The latency is far too great for a direct control approach, requiring the introduction of some levels of autonomy to the robots as well as pre-planning of actions. The robot being so far from home also has impacts on other aspects of its makeup. Batteries cannot be recharged through existing infrastructure like they could on Earth, and have to rely on solar energy or more long-term portable power sources. The robot is also unable to be repaired (outside of software solutions) once on the surface of Mars, placing restrictions on what operators dare to do with the robot, as any mistakes could mean the end of the mission (Ono, Fuchs, Steffy, Maimone, & Yen, 2015).

c. Solutions

Earth-guided robotic exploration of the surface of Mars places strict limitations on teleoperation systems, forcing a break from the traditional operator-robot direct control loop, through the introduction of some autonomy to the systems. The Mars Exploration Rovers (MER) for example are commanded once every Martian day with a series of tasks to execute. At the end of each day the robot is contacted again to retrieve the data and images collected during the day and the next day's activities are planned (Biesiadecki et al., 2007).

The inability to repair the rovers means that the operators have to be extremely conservative with their plans, and rely on automated processes onboard the rover to respond to the environment such as obstacle avoidance systems (Biesiadecki et al., 2007).

2.3.5 Analysis

It is apparent from the scenarios described above that the direct control teleoperation model described in Section 2.2.2 is not universally applicable.

Some scenarios such as Mars exploration simply cannot support it, as the latency introduced by the round trip time is so large that a human operator could not usefully operate the robot. Other systems such as robotic surgery still make use of direct teleoperation, but mediate the commands intelligently to ensure the safety of the patient by limiting potential operator mistakes. In both of these cases, some automation has to be introduced to the teleoperation system to allow the human operator to perform their task. The amount and type of automation is highly dependent on the application.

It is likely that most teleoperation systems could make use of and would benefit from some automation, although some systems such as the USAR robots described above shy away from it currently, as their deployment is in environments so complex that current automation techniques might prove more of a hindrance than help. This suggests that rather than one ideal model, a whole spectrum of modes for teleoperation might be more suitable. This is supported most notably by the work of Sheridan and Verplanck (1978) which began with two models: direct and supervisory control. These were later expanded on as shown in Figure 2.5, which showcases subcategories of those initial modes, as well as introduce full automatic control to the spectrum (Sheridan, 1992). The spectrum will be covered in more detail in Section 2.4

This section has shown that teleoperation is not limited to one model, with direct control only suitable for some teleoperation applications. There is an argument for the inclusion of automation in varying amounts to teleoperation systems, as dictated by the differing demands placed on them by the environments they are to perform in, as well as the limitations of current technology.

2.4 Automation in Teleoperation Systems

2.4.1 Overview

Section 2.3 introduced the idea of the teleoperation spectrum, a collection of models for teleoperation integrating differing amounts of automation. These

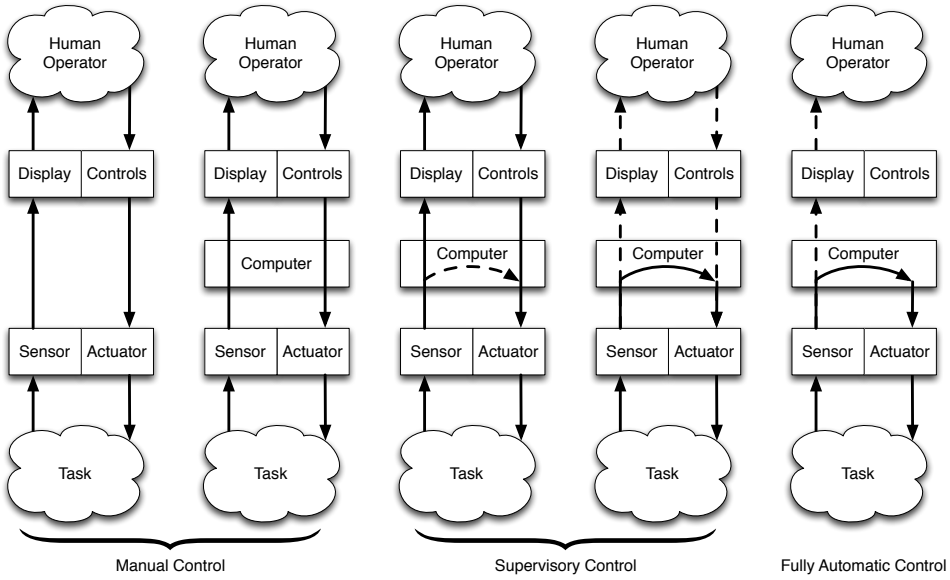


Figure 2.5: The spectrum of control modes (Adapted from Sheridan, 1992). Solid lines represent the primary control loop, with the dashed lines representing secondary influences. As the spectrum is travelled from manual control to fully automatic control, the role of the human operator reduces from sole operator to spectator, while the computer's (the automation) role does the opposite.

range from the automation-free direct control to full automatic control (See Figure 2.5). It is notable that as the inclusion of automation increases, the role of the human operator moves to the back seat, going from full control, to high-level commanding, to observer. This section considers the role of the human in teleoperation, discussing the need for human agency, the benefits and costs associated with the reduction of that agency, as well as expanding on the idea of the teleoperation spectrum.

2.4.2 The Spectrum of Control Modes

Figure 2.5 illustrates the traditional spectrum of control modes introduced by Sheridan (1992). This spectrum is comprised of five distinct control modes for teleoperation systems. These range from basic direct control (leftmost) to fully automatic control (rightmost), and are ordered left to right (least to most) by amount of automation present in the system.

The modes in the spectrum are distinguished by the way the control

loops in the system change as automation takes a bigger role in the control process. Sheridan also provides high level categories to categorise the modes: *Manual Control* (Two modes), *Supervisory Control* (Two modes), and *Fully Automatic Control* (One mode). These categories illustrate the changing role of the operator as we travel along the spectrum, transitioning from hands-on controller, through high-level commander, to passive observer.

This transition is reflected in the changes to the system's control loop: Manual Control modes use the traditional or basic control loop that directly connects operator and robot. The operator is firmly in the driver's seat, making both high-level decisions and executing low-level control actions. The second Manual Control mode introduces automation to the system (where before there was none), with a mediating computer present in the loop, providing assistance to the operator such as smoothing out unwanted signals from shaky hands, and inverse kinematic control. Guarded teleoperation interfaces (Pratt & Murphy, 2012) fit this description (although depending on implementations, they may span other control modes). In guarded teleoperation, the human operator is in control of the robot, but automated aids can restrict that control to avoid collisions with the environment for example. This can be useful for high-latency control scenarios, such as operation of Lunar rovers from earth (Gingras et al., 2014).

The Supervisory Control modes introduce a second control loop to the system that can take control of the robot. At this stage the operator is used as a high-level decision maker, commanding the mediating computer, which in turn issues the required low-level commands to the robot. The two Supervisory Control modes shown in the figure represent a shift in the human operator's responsibilities: the first requires input from the operator to function, the second accepts it, but does not necessarily require it. This represents the tipping point in the balance between operator and automation. Past these modes, the automation is responsible for most of the control. It is important to note, however, that this simply means that the bulk of the control work is done automatically; the motivations for that work and methods used to accomplish it are still in the hands of the operator. This is still somewhat true of the last mode on the spectrum,

Fully Automatic Control, as even though the automation requires no input from an operator, it is still performing tasks it is programmed to do. The operator is reduced to the role of spectator.

This spectrum does not describe every possible mode a teleoperation system could operate in, but rather highlights a series of small changes to one model, each step adding, modifying, or removing one element. It is also worth noting that the spectrum only describes the high-level dynamic between operator, automation, and robot, and does not specify in which manner these actors interact with each other. This makes the spectrum a useful tool to classify teleoperation systems, but of less use in actually designing them.

2.4.3 Humans and Automation

While the spectrum of control modes describes functional changes to teleoperation systems, it does not specify in any detail the effect these changes have on the role of human operators in the system. Research in other fields where automation has been introduced to control machines (such as aviation) gives some insight in how humans and automation interact.

Automated systems can and have replaced human workers in some jobs (which is a societal concern), but the introduction of automation does not always lead to their complete replacement. Rather, it changes the nature of the work they do. The role of humans in these system tends to switch from worker to supervisor when automation is introduced (Parasuraman & Riley, 1997). The automation of physical tasks has freed that side of human work, but automation that can fully replace humans remains both difficult to create and possibly undesirable (Bradshaw, Hoffman, Woods, & Johnson, 2013). Firstly, there is a perception that humans are more flexible, adaptable, and creative than automation (Parasuraman & Riley, 1997) (although more recent work has shown that mistrust in the abilities of automation may be reducing as automation becomes more prevalent in society (Bekier, Molesworth, & Williamson, 2011)). This has led to systems that attempt to automate most of a task, but still include a human operator

to take over in case of unexpected or hard to handle situations. Secondly, even "autonomous" systems are never really fully isolated. No system is capable of performing every task in every environment without help, and will at some stage need assistance, human or otherwise (e.g. for recalibration or repairs) (Bradshaw et al., 2013).

This teaming of humans and automation is not painless however. Parasuraman and Riley (1997) identify three categories of errors humans can make with automation, a categorisation that has driven much of human factors research since its publication (Lee, 2008):

- **Misuse**: the use of automation when it should not be used. Misuse of automation is exemplified in situations where the operator is overly reliant on the automation. This can be caused by overvaluation of the capabilities of the automation, or simply because of a failure to properly monitor the state of the automation.

- **Disuse**: not using automation when it should be used. Disuse of automation occurs when human operators purposely ignore available automation, such as blocking warning alarms and bypassing safety systems.

- **Abuse**: the deployment of automation by managers or designers without proper consideration for its impact on human operators.

When applied to the spectrum of control modes described in Section 2.4.2 and Figure 2.5, the concepts described above help establish what the human-automation interactions will look like as the spectrum is travelled. At the manual control end, the human operator is fully in control (effectively the human worker discussed above), as more and more automation is introduced into the system the role of the human operator shifts from "worker", through "supervisor", to "monitor" at the fully automated end. Regardless of the amount of automation present in the system, there is a role for the human operator to play, and the possibility of automation to be misused, disused, or abused.

2.4.4 Summary

This section described the effect the introduction of automation has on teleoperation systems. As Sheridan (1992)'s spectrum of control modes shows, Sheridan sees the introduction of automation as requiring two changes: i) a change in the system architecture to allow support for two sources of control input, and ii) a change in the role of the human operator. As the spectrum is travelled, from full manual control, to fully automatic control, the role of the human operator and the automation effectively switch.

While in full manual control there is no automation, fully automated control still retains a human presence, even if it is only in a monitoring role. There are still limitations in the capabilities of automated systems, and in the complex environments teleoperated robots usually operate, these systems will encounter situations they are not programmed to deal with efficiently or at all. When these situations arise, the human operator can step in and rectify any issues. The cohabitation of humans and automation is not without its fair share of issues however, as improper use of automation, either by the operator (misuse or disuse) or by the designers of the system (abuse), can affect the performance of the system and cause unwanted damage.

2.5 Implementations of Teleoperation

2.5.1 Overview

The previous sections have introduced the basic formula of teleoperation as well as its more complex forms where human operators work hand in hand with automated systems. As shown in Section 2.3, the form a teleoperation system takes is largely dependent on the restrictions placed on it by the robot's environment as much as the task it has to perform. This section presents six implementations of teleoperation systems, comparing them and classifying them using five criteria which encompass both the operating restrictions placed on them as well as the solutions they present. The criteria described below were chosen as they represent restrictions the creators of these systems had to contend with when designing them, as

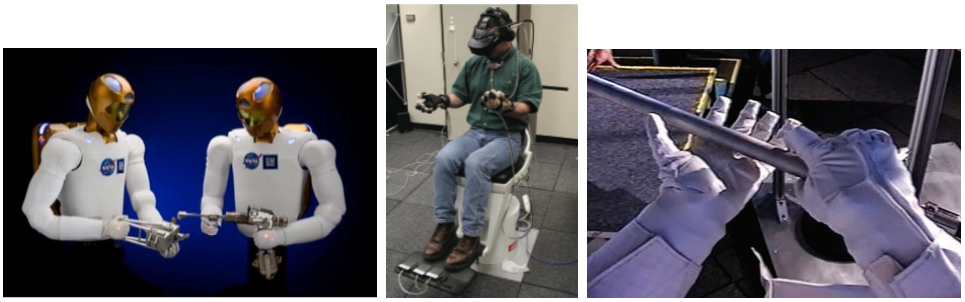
well as their solutions.

- Role of the system: What is the main use for the system? Is it replacing humans or performing a new task that humans could never do?
- Communications: What are the environmental restrictions placed on communications? (Distance, Interference, etc.)
- Control mode: Where does the system fall on the spectrum of control modes?
- Neglect tolerance: How long can the system go without human input before reaching a failure state?
- Attention demand: How much human attention does the system need to operate normally?

2.5.2 NASA's Robonaut

The Robonaut was developed by NASA to assist astronauts during Extra-Vehicular Activities (EVA), and other space-related activities (Diftler et al., 2010). This robot is anthropomorphic, designed as a copy of a human torso arms and head, as shown in Figure 2.6a. The Robonaut can be attached to a variety of mounts to allow it to perform tasks in various environments (Goza et al., 2004). The current generation of the robot, Robonaut 2, can be operated using two methods. The first method uses a high-level programmable interface (Diftler et al., 2010). The second method uses a telepresence-based interface that uses the anthropomorphic shape of the robot to maximum use, by directly slaving the robot to the movements of the operator (Bluethmann et al., 2003), the version of this interface used for Robonaut 1 is shown in Figures 2.6b and 2.6c.

Robonaut 2 is now present on the International Space Station, from which it can be operated by the astronauts on-board, either in its autonomous mode or its telepresence mode, or from the ground in its autonomous mode (Diftler et al., 2012). The telepresence approach is only possible on-board because the increased time delays of long distance



(a) **(b)** **(c)**

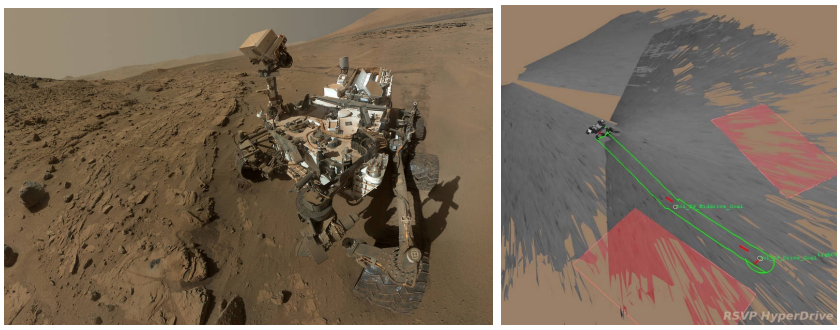
Figure 2.6: The Robonaut robot (a) (Diftler et al., 2010), telepresence gear (b) and operator view (c) (Rehnmark et al., 2005).

operation would render this telepresence-based teleoperation impossible (Bluethmann et al., 2003). This direct mapping between the robot and the operator comes with multiple advantages, as the operator feels like Robonaut is an extension of their own bodies, which decreases the need for operator training (down to five minutes in some trials (Ambrose et al., 2000)) and maximises performance. The use of the interface does come at a cost, as the physicality of the control scheme tires operators during prolonged use (Bluethmann et al., 2003). While this operation mode was favoured for Robonaut 1, Robonaut 2 is intended to be primarily operated in its autonomous mode, with the telepresence interface a fall-back (Diftler et al., 2012).

2.5.3 Curiosity

The Mars Science Laboratory rover Curiosity (Shown in Figure 2.7a), on the surface of Mars since August 2012, is tasked with evaluating the planet's habitability (Greicius, 2015). Due to the vast distance between Curiosity and its control station on Earth, Curiosity has to be commanded asynchronously. Its instructions sent in the morning, with the results of its operations sent back in the evening. The next day's activities are then planned and sent to Curiosity the following morning (Venkatraman, 2015).

Curiosity's movements are planned using the Rover Sequencing and Visualisation Program (RSVP) suite of tools, which consists of the Robot Sequence Editor (RoSE), HyperDrive, and ImageBrowser. RoSE is used to



(a) Mars Science Laboratory Curiosity (Greicius, 2015)

(b) Sequence Simulation Using HyperDrive (Wright, Hartman, Maxwell, Cooper, & Yen, 2013)

Figure 2.7: Curiosity and path planning interface.

create command sequences, HyperDrive for simulation and rehearsal, and ImageBrowser for the visualisation of images taken by Curiosity (Wright, Hartman, Maxwell, Cooper, & Yen, 2013). Paths are created and tested in simulations (See Figure 2.7b) until deemed satisfactory and sent to the rover.

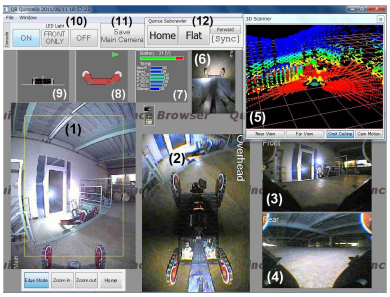
This movement planning is a tedious task, as the planners have to consider the terrain the rover needs to traverse. Rocks driven over can cause damage to the wheels, and soft soil can bog the rover, and are therefore carefully avoided. A sol's (Martian day) travel, which ranges from 30 to 70m can take a team of three people eight to ten hours to plan (Ono et al., 2015).

2.5.4 Quince

The Quince robot was originally designed for use in post-disaster search and rescue missions. After the Fukushima Daiichi nuclear power plants suffered damage caused by an earthquake and tsunami, the Quince was updated to perform missions in a highly radioactive environment (Nagatani et al., 2013).

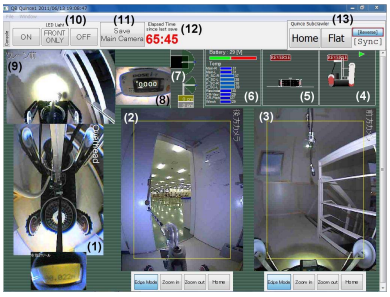
Due to the working conditions inside the reactor, the Quince could not be teleoperated wirelessly from the outside, as the thick walls of the reactor would block radio signals. To overcome this limitation, two Quinces were put into service: a wired Quince that spooled a communications cable

(a) Quince: wireless exploration robot.



(b) Exploration robot UI.

(c) Quince: cabled robot.



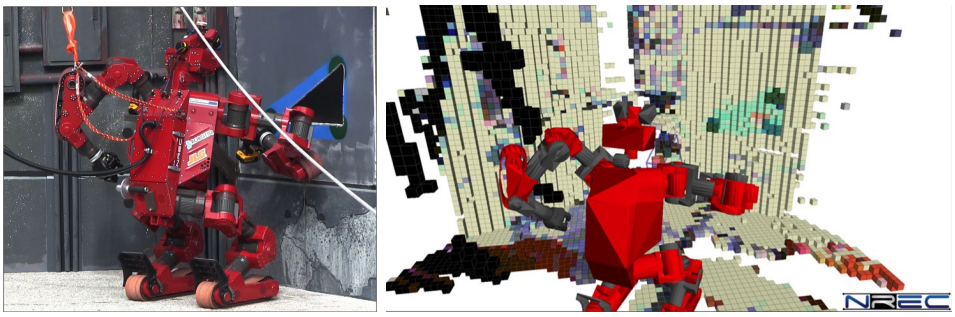
(d) Cabled robot UI.

Figure 2.8: The Quince dual robot system. One robot has cabled communications and broadcasts a wireless network for the other. Both robots are outfitted differently and the interfaces reflect these differences. The exploration robot has access to a laser scanner (output displayed top right of the screen), while the cable robot is equipped with a crane and dosimeter both of which have associated cameras feeds (top and bottom left of screen respectively) (Nagatani et al., 2011).

behind it (See Figure 2.8c), and a wireless Quince (See Figure 2.8a) that was controlled through a network broadcast from the cabled robot. Each robot was controlled independently by a different operator using the interfaces shown in Figures 2.8b and 2.8d (Nagatani et al., 2011).

2.5.5 CHIMP

CHIMP is a humanoid robot that participated in the DARPA Robotics Challenge (DRC), placing third in the finals (DARPA, 2015). As a humanoid robot, CHIMP has two legs, but rather than use these for bipedal locomotion, privileges the use of motorised tracks mounted on its feet (and arms if the situation calls for improved stability) to drive around (Stentz et al., 2015).



(a) CHIMP performing the wall task at the DRC Trials (Stentz et al., 2015)

(b) Rendered view of CHIMP's environment (Stentz et al., 2015)

Figure 2.9: The CHIMP robot and situational awareness assistance display.

CHIMP's operator interface displays video and telemetry from the robot, as well as a 3D display of the robot's environment. This display uses data from CHIMP's sensor head (laser rangefinders for 3D mapping, cameras for colour information) and knowledge of its own position to render an accurate model of CHIMP in its operating environment (Stentz et al., 2015). This display provides the operator with situational awareness not available in most teleoperation scenarios, where the operator cannot see the robot they are controlling directly.

CHIMP's operators can command the robot in three distinct modes: task mode, workspace mode, and joint mode. In task mode, the operator gives high-level instructions to the robot, such as which object to grasp, how to grasp and how the objects are able to move. The system then plans the motions the robot will need to take. In workspace mode, manual control is returned to the operator, who commands the position of the limbs, with the system computing joint positions to suit. Finally, joint mode returns full control to the operator, who controls the position of individual joints (Stentz et al., 2015).

2.5.6 DaVinci Surgical Telemanipulator

The DaVinci telemanipulator is used in hospitals around the world to perform keyhole surgery. Unlike many other teleoperation applications, the DaVinci is not needed to remove operators from harm's way, but to decrease danger to the patient being operated on. The system enables

Figure 2.10: Operation room with the DaVinci surgical telemanipulator (Melvin et al., 2002).

surgeons to perform better than they could operating traditionally (Melvin et al., 2002).

The DaVinci is operated by the surgeon through an immersive interface: the robot is slaved to the motions of controllers that the surgeon grasps, a system of pedals, and three-dimensional video feedback is provided by a stereo viewer (Palep, 2009). This system enhances the capabilities of surgeons by allowing them to perform finer operations through the use of motion scaling, tremor suppression, allowing non-rigid instruments to be used, as well as improving visualisation (Marohn & Hanly, 2004)

The surgeon is not expected to operate alone, and is assisted by an operative team. The DaVinci system includes an additional monitor for use by an assistant surgeon, as well as an intercom system allowing the surgeon to communicate with his team while still looking into the viewer (Palep, 2009).

2.5.7 Autonomous Mining Vehicles

To both increase productivity and keep workers out of dangerous areas, the mining industry has been seeking to automate the heavy machinery that operates on its mine sites. A significant portion of this automation happens through the retrofitting of existing trucks (See Figure 2.11a). These trucks

(a) Autonomous haul truck. **(b)** Operator supervising multiple trucks.

Figure 2.11: Autonomous mining truck and control station (Brown, 2012).

are fitted with drive-by-wire systems, position sensors (GPS), and obstacle detection sensors. They use these sensors to travel pre-set paths, avoiding collisions on the way (Brown, 2012).

Human operators are still involved in the control of the machines at a high level, as they set high-level goals for the trucks to follow. A single operator can supervise multiple trucks (See Figure 2.11b). The automation will then use those goals to coordinate the activities of multiple trucks, avoiding collisions and deadlocks (Brown, 2012).

The nature of the transition from human drivers to autonomous trucks is entirely driven by business decisions. Some mines will take an incremental approach, first manually teleoperating trucks then slowly including more and more automation, while others such as Rio Tinto's Mine of the Future in Western Australia are pushing straight for autonomy (Brown, 2012).

2.6 Discussion

The teleoperation implementations shown in Section 2.5 showcase a wide range of applications for teleoperated robots and solutions to the problems inherent with teleoperation in varied environments. As shown in Table 2.1, most of the implementations described above are positioned in the "manual control" section of the spectrum of control modes. Manual control is still the most widely used control paradigm due to its inherent robustness, human operators being able to think and use the robot creatively, as well as the

Table 2.1: Classification of five teleoperation system.

System	Role	Communications	Control Mode	Neglect Tolerance	Attention Demand
Robonaut 2	Replacing humans	Deployment specific low-latency links	Manual	Low	High
Curiosity	Going where humans cannot go	Interplanetary radio link	Supervisory	High	High
Quince	Going where humans cannot go	Hybrid wired/wireless dual robot system	Manual	Low	High
CHIMP	Competing in the DARPA Robotics Challenge	Local links (Restricted by DARPA during trials)	Varies: Manual to Supervisory	Low	High
Da Vinci Surgical Robot	Enhance Human Capabilities	Cabled (Operator and robot within same room)	Manual	Low	High
Automated Mining Trucks	Replacing humans	Radio Network	Supervisory	High	Low

difficulties associated with implementing robust automation for real tasks. The supervisory control implementations are seen in situations where the introduction of automation is easier (autonomous mining trucks), required (Curiosity), or a fallback to manual control is available (CHIMP). It is clear that the state of the art in automation is currently not sufficient to allow robots to operate autonomously outside of the tasks they are specifically designed for. For the time being, manual control is likely to be the norm for robots engaging in complex tasks or for use in non-specialised scenarios. Manual control is far from ideal, and as shown in Section 2.2.3, it is extremely taxing on the operators, and thus should be seen as undesirable if necessary, not as the norm.

With manual control undesirable and automation not capable enough, a compromise could provide a better answer. However, the supervisory control implementations shown in Section 2.5 reveal that supervisory control faces similar limitations to full automation: it is mostly used to release

specific autonomous behaviours in specialised scenarios, rather than as a more general problem-solving solution.

The current ideal is more likely to be similar to interfaces such as CHIMP's (See Section 2.5.5), which allow the operator to move up and down a spectrum of control modes during a task's execution. This type of control is known as adjustable autonomy. Adjustable autonomy enables the operator to rely on automated or semi-automated components to perform control tasks when able and only have to manually control the robot in situations the automation cannot handle. This means the robot can be used more generally while still attempting to relieve the operator of some workload. The automation becomes a tool just like any other that the operator can deploy when they feel appropriate, possibly in ways its creators had not considered. Adjustable autonomy interfaces show great promise, and are investigated in detail in the next chapter.

2.7 Summary

This chapter reviewed the current state of teleoperation, from a theoretical view through to practical implementations. While within the greater subject of robotics the robots themselves tend to be the focus of both attention and advances, this chapter has shown that improvements in end-to-end robot control systems are much needed to expand their capabilities. Section 2.2 introduced the basic workings of teleoperation, as well as the challenges teleoperation systems have to overcome. Section 2.3 showcased different use cases for teleoperation systems, highlighting the restrictions each place on those systems. These restrictions occasionally require the introduction of automated components to allow teleoperation, which was expanded on in Section 2.4, where Sheridan's spectrum of control modes was described. The spectrum provides a way of classifying teleoperation systems based on the amount of automation present as well as architectural differences. This section also discussed the interaction between humans and automated systems. Finally, Section 2.5 provided a look at ways teleoperation systems have been implemented, information which was used in Section 2.6 to draw conclusions about the current state

of teleoperation as well as where the next likely progress in the area will come from.

Chapter 3

Adjustable Autonomy

3.1 Overview

Capable, fully automatic control of robots is a desirable end goal, albeit one currently infeasible for many scenarios. In the short term, the merging of automation techniques and teleoperated control promises more compelling results. This is as a result of the complimentary relationship of automated and teleoperated control. The lack of flexibility and environmental understanding displayed by automated control systems can be compensated for by the inclusion of a human operator in the loop (Dorais, Bonasso, Kortenkamp, Pell, & Schreckenghost, 1999). Conversely, routine repetitive tasks are usually easily automatable (Autor, Levy, & Murnane, 2001), but tend to challenge human attention and consistency.

This chapter presents an analysis of the current state of *adjustable autonomy* systems. These are semi-autonomous systems that allow the amount of autonomy they use to be reduced or increased during operations. Such systems are thought to bridge the gap between human control and fully automated control through the selective use of automation. Adjustable autonomy allows the operator to take back manual control in case of a failure of automation or simply when no automation is available for the task.

The remainder of this chapter is structured as follows. Section 3.2 defines and introduces the theory behind adjustable autonomy, detailing

arguments for its use, and explaining the levels of autonomy an adjustable autonomy system might use. Section 3.3 compares the strategies employed by real systems that make use of adjustable autonomy to manage these changes. These systems are compared and the next step forward for adjustable autonomy systems is considered in Section 3.4.

3.2 Adjustable Autonomy

3.2.1 Overview

Adjustable autonomy was originally designed to allow human controlled systems to deal with the long latency of extra-terrestrial teleoperation, it is defined as the capability of an autonomous system to have the amount of autonomy (as opposed to human control) it uses changed during its operation (Dorais et al., 1999; Goodrich, Olsen, Crandall, & Palmer, 2001).

Dorais et al. (1999) suggest that adjustable autonomy is essential in scenarios where autonomous systems might encounter difficulties in their task accomplishment. With full automation not providing the capabilities needed to operate many complex systems, adjustable autonomy provides a suitable if temporary alternative, as it is expected to increase the capabilities of autonomous systems, decrease time and money spent on automation development, and lead to an improvement in system reliability as well as an increase in user understanding, trust and control of the system (Bradshaw et al., 2004; Ball & Callaghan, 2012). The remainder of this section will discuss adjustable autonomy, explaining the reasoning behind its use, and describing its potential applications as well as its advantages over human operation or full automation.

3.2.2 Benefits of Adjustable Autonomy

To mitigate the added complexity that the support for multiple operating modes comes with, adjustable autonomy must provide sufficient benefits. This section details four qualities of adjustable autonomy systems, all of which revolve around reducing difficulties associated with traditional teleoperation systems and/or fully automated systems.

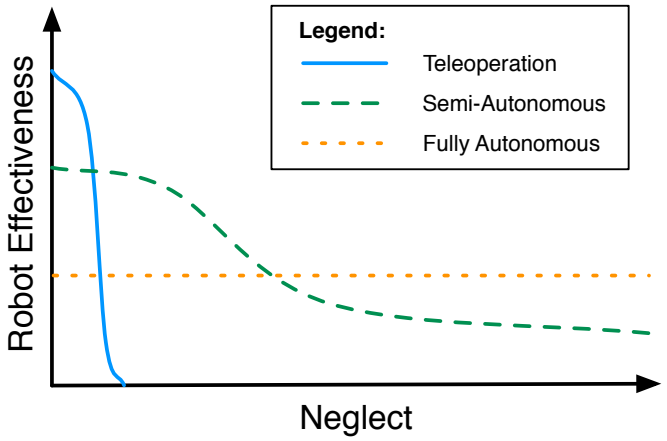


Figure 3.1: The neglect curve (From (Goodrich, Olsen, Crandall, & Palmer, 2001)).

a. To Support Neglect

The neglect graph (Shown in Figure 3.1) was introduced by Goodrich et al. (2001) to illustrate the effect of operator neglect on a robot's effectiveness at accomplishing the task it is undertaking. Though the term usually has negative connotations, it is considered here to be a positive for operators. Neglect can be characterised as the time since the operator last interacted with the robot's interface (Olsen & Goodrich, 2003). As neglect increases, robot effectiveness tends to decrease in systems that require human input. The rate of this decrease is related to the amount of automation present in the system as well as the application: flying machines will tend to be at risk faster from such a decrease than their earth-bound counterparts for example. The support of neglect is particularly important in scenarios where operators need to divide their attention, such as when operating several robots (Olsen & Goodrich, 2003).

The automation needed to support neglect may tend to lead to a reduction in the breadth of tasks the system can accomplish. Adjustable autonomy allows the support of neglect while limiting this effect; operators can potentially switch to a more automated mode when they need to neglect the robot, and later resume manual control with its higher attention demands (Goodrich et al., 2001).

b. To Manage Operator Trust

Trust has a role to play in the deployment of automation, as discussed in Section 2.4. Workers may not trust an automated system they think is unreliable for example. Bradshaw et al. (2004) suggest that adjustable autonomy has a role to play in easing the integration of automated systems in workplaces. They identify two driving forces that regulate the use of automation: *convenience* and *comfort*. Convenience is the drive to automate as much of the task as possible to relieve oneself of having to do it, while comfort limits this due to distrust in the capabilities of the automated module taking over the task. The authors also note that in some cases, operators may want to limit automation due to the enjoyment they get out of the work, citing the example of skilled drivers preferring to drive manual over automatic transmission cars.

Adjustable autonomy can help balance these two drives by allowing some flexibility in the integration of the automation, only delegating tasks to the automation that it can be trusted to accomplish (Falcone & Castelfranchi, 1999). By placing boundaries on the degree of autonomy a system has at any stage of a task's execution, an operator should be be satisfied that as much of a task is being performed automatically as possible while not allowing the automation to initiate actions during segments it is not trusted in (Bradshaw et al., 2004).

c. To Improve Resilience

A resilient system is a system that can remain robust when faced with disturbances (Zieba, Polet, Vanderhaegen, & Debernard, 2010). This resilience happens through three processes: preventing disturbances, coping with ongoing disturbances, and recovering from disturbances after the fact. Each of those processes require adaptability from the system.

Zieba, Polet, Vanderhaegen, and Debernard (2008) propose that adjustable autonomy can improve a system's resilience, by improving the adaptability of automated systems. By altering the task allocation between human and automation, the system can call upon either's agent capabilities to deal with the disturbance, either by assisting the other, or

fully taking over. An example of this would be an automated system with the capability of detecting a disturbance but without the skill to deal with it. This system could trigger a change in autonomy, requesting assistance from the human operator. It is worth noting that this runs in opposition to the idea of using adjustable autonomy to maximise neglect. If an operator sets a system to an automated mode to take a break, they will not be there to respond to calls for help.

d. To Support Long Term Deployments

Dorais et al. (1999) and Kortenkamp, Keirn-Schreckenghost, and Bonasso (2000) suggest that adjustable autonomy systems are required in many facets of long space missions. Life support systems for example should be fully automated, but allow human intervention in particular cases such as during maintenance. A team in orbit around Mars could operate a robot on the surface for exploration tasks, but is unlikely to be controlling the robot manually for days on end. It is more likely they would give it high-level commands and let it operate semi-autonomously for the majority of the time, only taking over when direct involvement is required, to limit the time and attention they would need to devote to it.

In addition to the long duration of the mission, the isolation of the participation also has an impact on the use of automation. When an autonomous system on Earth malfunctions, repairs are available readily, while a team on Mars would have no such luxury. Allowing the level of automation to be changed in response to unforeseen circumstances is as much a safety measure as a convenience (Sierhuis et al., 2003). The opposite is also true, the automation may need to intervene during manual control of a system, to safeguard that system against operator errors, especially potentially fatal ones (Dorais et al., 1999).

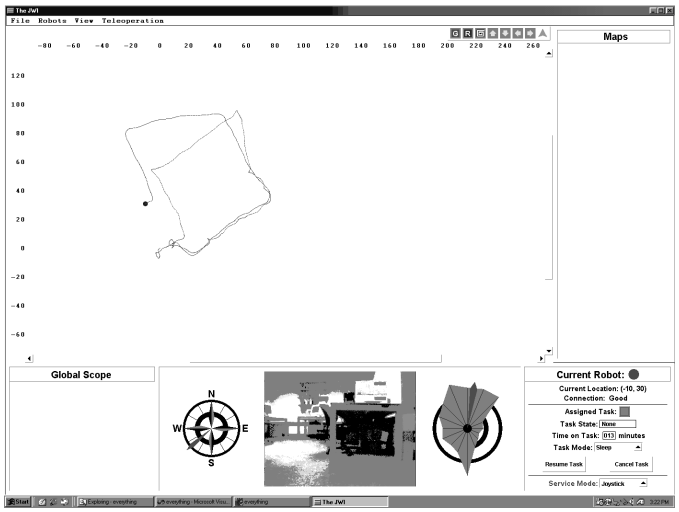


Figure 3.2: The interface used by the operators in Goodrich, Olsen, Crandall, and Palmer (2001)'s adjustable autonomy system.

3.2.3 Levels of Autonomy

To modify the amount of automation in their control, adjustable autonomy systems make use of *levels of autonomy*[1], operating modes that define the involvement of automation and human operators (Dorais et al., 1999). For example, Goodrich et al. (2001) designed and adjustable autonomy system to support operator neglect (See Section 3.2.2) allowing operators to divide their attention between two Nomad SuperScout robots. The interface is shown in Figure 3.2, displaying a map of the path travelled by the robot, as well as sensor information from the robot underneath. Four discrete levels were used in this implementation: full autonomy, goal-biased autonomy, waypoints and heuristics, and intelligent teleoperation. Levels of autonomy are set manually by the human operator using the robot control interface. These levels are described in detail below.

- **Full Autonomy**: No human input, the automation is in full control. This level is of limited overall usefulness as the automated capabilities are limited to the exploration and mapping of the environment the robot is in. This level is however highly tolerant to neglect.

[1]This term is used interchangeably with control/operating modes in the literature, and will be used the same way in this chapter based on the source's wording.

1. Computer offers no assistance; human does it all.
2. Computer offers a complete set of action alternatives.
3. Computer narrows the selection down to a few choices.
4. Computer suggests a single action.
5. Computer executes that action if human approves.
6. Computer allows the human limited time to veto before automatic execution.
7. Computer executes automatically then necessarily informs the human.
8. Computer informs human after automatic execution only if human asks.
9. Computer informs human after automatic execution only if it decides too.
10. Computer decides everything and acts autonomously, ignoring the human.

Figure 3.3: Levels of Autonomy (From (Goodrich & Schultz, 2007)). Each level describes the balance of work between a human operator and automation a system could have.

- **Goal-Biased Autonomy**: The robot is still mostly autonomous, but its goal selection is biased by the human operator's use of goal/risk icons. This level is less tolerant to neglect, as the robot stops when the biased goal is reached.
- **Waypoints and Heuristics**: The human operator sets waypoints and heuristics for the robot. The operator can specify waypoints, that attract the robot, obstacles, that repulse the robot, and heuristics, that constrain the robot's movement dirtectionally. This level requires more human input, trading a reduced tolerance to neglect for better task efficiency.
- **Intelligent Teleoperation**: The robot is operated by constant human input through the use of a joystick. The robot interprets the commands and executes them in a way that compensates for latency related issues. The system includes some safe-guarding to prevent the operator from colliding with obstacles (similar to help found in guarded teleoperation systems (Pratt & Murphy, 2012)), but cannot initiate actions, greatly limiting its tolerance to neglect.

The idea of levels of autonomy predates adjustable autonomy systems,

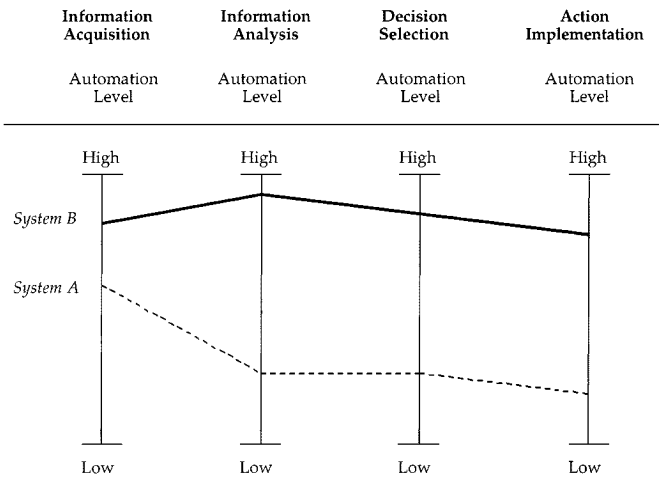


Figure 3.4: Characterising two systems' levels of automation by rating the level of automation at each major function (Parasuraman, Sheridan, & Wickens, 2000).

with an early example of levels of autonomy proposed by Sheridan and Verplanck (1978) (an updated version of these levels adapted by Goodrich and Schultz (2007) is shown in Figure 3.3). While levels of autonomy are more commonly used to described the particular modes of operation of a particular system, Sheridan and Verplanck (1978)'s levels try to imagine the full range of modes that could exist between full human control and full automation.

Later work found this scale to be somewhat limited, only describing the interaction between a human operator and an automated decision-making system (Parasuraman, Sheridan, & Wickens, 2000). Parasuraman et al. (2000) proposed an enhanced model that aims to provide a more accurate characterisation of levels of autonomy by breaking down the potential tasks a system might have to do into four major functions and applying the levels presented in Figure 3.3 to each of those categories rather than to the system as a whole. Figure 3.4 illustrates how two systems might be compared using this method. Each has a characteristic profile of automation levels.

Parasuraman et al. (2000) split the task of operating a system into four major functions: i) Information acquisition, ii) Information analysis, iii) Decision selection, and iv) Action implementation. Rating each of these functions for a particular system gives a clearer picture of the role the

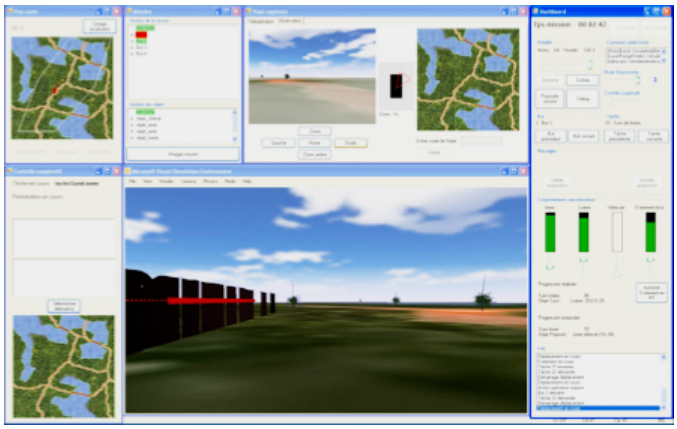


Figure 3.5: Zieba et al.'s adjustable autonomy user interface (Zieba, Polet, & Vanderhaegen, 2011). This interface allows the control of a virtual robot in a simulated environment. The level of autonomy is adjustable through the use of a slider on the left of the screen. Goals are displayed at the top of the screen and can be navigated through by the user using buttons.

automation and the human operator have to play in how the system functions. While the authors are still only talking about characterising whole systems, this way of characterising levels of autonomy is applicable to the levels used in an adjustable autonomy system.

Zieba et al. (2010) make use of this classification to describe their adjustable autonomy system used to control a virtual robot in a simulated environment (Pictured in Figure 3.5). Table 3.1 shows this classification. Their system makes use of four levels of autonomy, which they call modes zero through three. Mode zero has everything set to full human control, mode one sees the human operator move to a supervisory role, being primarily responsible for analysing information and making decisions, mode two mostly removes the human operator from a control role, only asking for advice on decision-making, with the ability to proceed if the operator does not respond. The final mode is fully autonomous.

These modes show the importance of Parasuraman et al. (2000)'s major function breakdown, as the progression between these levels is not linear from full human control to full automation, and could not be accurately characterised using a more broad brush approach. While Zieba et al. (2010) are not specific about why their levels progress in this way, some of the

motivations discussed in Section 3.2.2 might provide some insight. The issues with trust in automation described by Bradshaw et al. (2004) could be responsible, as we see both information analysis and decision selection as the last functions to be fully automated. In contrast, information acquisition and action implementation are automated from the second mode onwards. It is possible that these last two automated functions perform better overall and are more trusted to accomplish tasks set to them, while the other two are less capable and only given free reign when the task complexity is low enough for them to be effective.

The levels in an adjustable autonomy system are likely to be unique to that system, based on the needs of the system and the capabilities of the automation the system uses. The literature on levels of autonomy, such as that which is discussed above, is not prescriptive, but rather better used descriptively as a common tool for researchers or industry to classify their systems. This need for classification was recognised by the National Institute of Standards and Technology (NIST) that produced the Autonomy Levels for Unmanned Systems (ALFUS), a tool for the evaluation of a system's level of autonomy (Huang, Pavek, Novak, Albus, & Messin, 2005). Having a tool allowing the comparison of these potentially very different systems is useful for both creators and purchasers of systems. The former can use this common ground to compare their product to others, allowing them to improve where they fall short, and the latter to have a consistent measuring tool to find the product that best fits their needs.

Table 3.1: Classification of Zieba, Polet, Vanderhaegen, and Debernard (2010)'s adjustable autonomy robot control system (Adapted from Zieba, Polet, Vanderhaegen, and Debernard (2010)). H marks tasks that are the responsibility of the human operator, A of the automation.

Function	**Mode 1**	**Mode 2**	**Mode 3**	**Mode 4**
Information acquisition	H	A	A	A
Information analysis	H	H-A	A	A
Decision selection	H	H-A	A-H	A
Action implementation	H	A	A	A

3.3 Changing Levels of Autonomy

3.3.1 Overview

Goodrich et al. (2001) refer to the setting of the level of autonomy in an adjustable autonomy system as a *meta-control task*, which could itself be theoretically done either by the human operator, automatically, or be a shared responsibility. While early implementations of adjustable autonomy systems only had humans in charge of setting the level of autonomy (Goodrich et al., 2001), some more modern implementations support autonomy changes that include automated triggers (Zieba et al., 2010). Where this level-change responsibility lies, and whether or not that change is planned, is likely to have a significant impact on the operation of that system. For example, a system that automatically adjusts its level of autonomy must be able to cope with the human operator not being ready, either through reaching out to the operator via messaging if there is an operator "on call", or by settling down into a safe state until someone can attend to it.

This section presents a survey of six implementations of adjustable autonomy systems, focusing on where the meta-control authority lies, as well as the nature of those changes: planned or dynamic. For each of these implementations, a representation of the levels of autonomy used by the system is provided using the model suggested by Parasuraman et al. (2000).

3.3.2 Goodrich et al. (2001)

The system presented by Goodrich et al. (2001) is designed to allow the human operator to control two robots to perform a series of tasks. Using the one interface, the operator can switch between the robots, controlling them one at a time, and setting appropriate levels of autonomy for each robot based on the attention the operator can give it. The system offers the operator four levels of autonomy, ranging from *full autonomy* to *intelligent teleoperation* (See Table 3.2).

As the system is designed to support operator neglect, in all of the

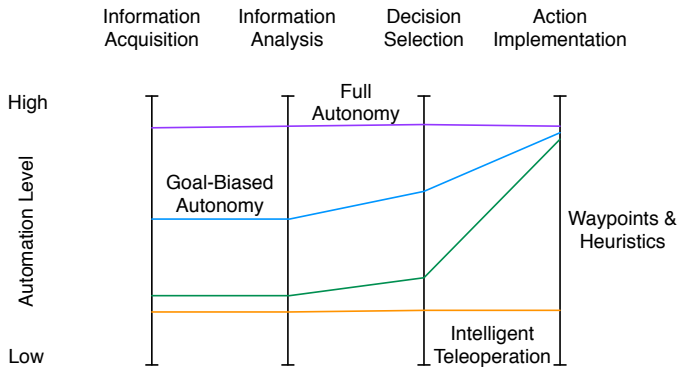


Figure 3.6: Characterisation of the Levels of Autonomy used by the system presented by Goodrich, Olsen, Crandall, and Palmer (2001).

Table 3.2: Levels of Autonomy and adjustment strategy for the system presented by Goodrich, Olsen, Crandall, and Palmer (2001).

Level Adjustment		**Levels of Autonomy**
Responsibility	**Trigger**	
Human	Dynamic	Full Autonomy
		Goal-Biased Autonomy
		Waypoints and Heuristics
		Intelligent Teleoperation

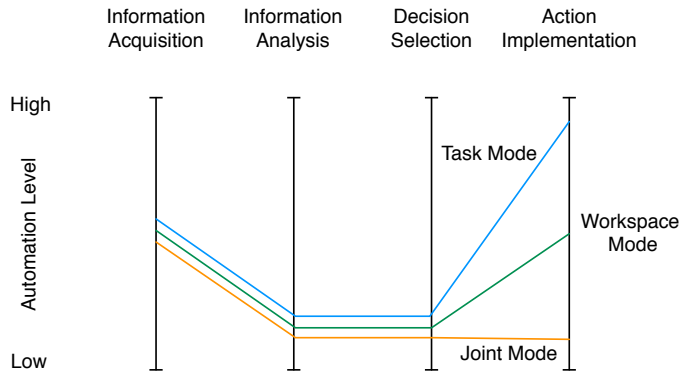


Figure 3.7: Characterisation of the Levels of Autonomy used to control CHIMP.

levels available except intelligent teleoperation, the robot is capable of remaining operational while the operator attends to the other robot. Goodrich et al. (2001) note that as the levels get closer to full autonomy, the capabilities of the robot decrease, but the robot can be neglected for longer (see Figure 3.1). Setting the level of autonomy for a robot becomes a balancing act between needing to attend to the other robot and keeping this robot achieving meaningful goals.

The levels of autonomy are set by the human operator, a responsibility that is crucial to the functioning of the system. In many systems, such as the control system for the CHIMP robot described in the next section, the levels of autonomy are mostly used to ease the work of the human operator. In this system, choosing the appropriate level for each robot is essential to keeping both robots running smoothly by properly managing operator neglect.

3.3.3 CHIMP (Stentz et al., 2015)

The CHIMP robot's teleoperation interface (described in more detail in Section 2.5.5) makes use of adjustable autonomy. The levels of autonomy used by the system are very human-centered, apart from the collection and merging of sensor data, which is done autonomously (see Figure 3.7). The most automated level is *Task Mode*, in which the human operator can highlight objects in the environment so CHIMP can interact with them.

CHIMP's levels are heavily focused on reducing the difficulty of action

Table 3.3: Levels of Autonomy and adjustment strategy for CHIMP's control system.

Level Adjustment		Levels of Autonomy
Responsibility	**Trigger**	
Human	Planned and Dynamic	Task Mode
		Workspace Mode
		Joint Mode

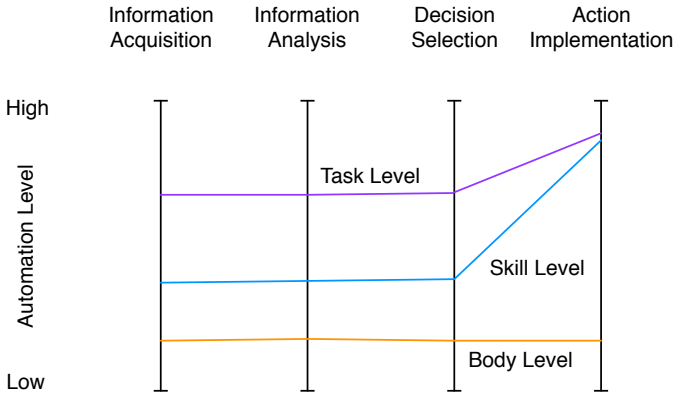


Figure 3.8: Characterisation of the Levels of Autonomy used to control Cosero and Dynamaid.

implementation, with each level only altering that part of the control task, and level selection being based solely on the tradeoff between speed and capability of the automation (Stentz et al., 2015).

CHIMP's levels of autonomy are under the operator's control at all times, allowing the operator to select the level most appropriate to the task at hand (Stentz et al., 2015). Due to the majority of the DARPA Robotics Challenge tasks being known in advance, and the teams having the opportunity to test their robots on the trial tasks ahead of time (Stentz et al., 2015), it is likely that the operators knew which mode was best for a particular task, and that outside of emergencies, changes in levels were planned. Similar approaches were used by two other DRC teams: team VIGIR (Kohlbrecher et al., 2015), and team THOR (Yi et al., 2015).

Table 3.4: Levels of Autonomy and adjustment strategy for Cosero and Dynamaid.

Level Adjustment		Levels of Autonomy
Responsibility	**Trigger**	
Manual	Planned	Task Level
		Skill Level
		Body Level

3.3.4 Cosero and Dynamaid (Muszynski, Stuckler, & Behnke, 2012)

The interface Muszynski et al. (2012) created for their Cosero and Dynamaid robots is a tablet-based adjustable autonomy system which allows the human operator to control these robots using three levels of autonomy *task level*, *skill level*, and it body level. Both robots possess a library of skills they can perform autonomously or semi-autonomously which is used in task and skill levels to relieve the operator of workload. Task level autonomy allows the human operator to specify high-level goals by providing the robot with contextual information from a limited pool such as the actions to perform and the locations to perform them in. Skill level autonomy is more directly based around this skill library, as the operator chooses a skill to use and provides the robot with information relating to the execution of that skill, such as marking up an object to grasp. Body level operation is a low-level control mode which allows the operator to use the robot for tasks it is not programmed to execute. This level still has some autonomy however as the robot can prevent the operator from colliding with objects (Muszynski et al., 2012).

In this interface, the changes in levels of autonomy are solely under the control of the human operator. Because of the skill-driven nature of these robots, changes in autonomy are used to make up for capabilities that have not been automated yet. Changes in levels are predictable as long as the task undertaken is known ahead of time.

3.3.5 Côté, Canu, Bouzid, and Mouaddib (2012)

Côté et al. (2012) present a robot control system that coordinates multiple operators and multiple robots. The robots operate autonomously until they

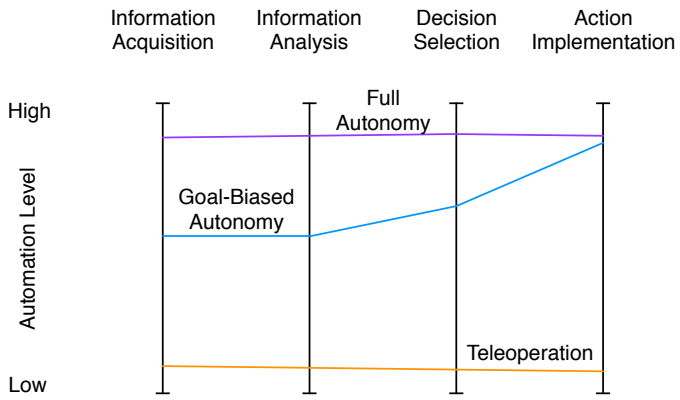


Figure 3.9: Characterisation of the Levels of Autonomy used by the system presented by Côté, Canu, Bouzid, and Mouaddib (2012).

Table 3.5: Levels of Autonomy and adjustment strategy for the system presented by Côté, Canu, Bouzid, and Mouaddib (2012).

Level Adjustment		**Levels of Autonomy**
Responsibility	**Trigger**	
Mixed	Dynamic	Full Autonomy
		Goal Biased Autonomy
		Teleoperation

encounter a situation their automation cannot handle, at which point one of the operators is assigned to them to help. The system makes use of three levels of autonomy to allow this to happen (As shown in Table 3.5). The robots operate under *full autonomy* until they encounter an issue, at which stage they asses the criticality of the issue before requesting help. The system then calculates the expected time the request will take to be dealt with as well as the expected gain from the request being fulfilled. The best request (calculated using gain/time - assuming multiple requests are sent simultaneously) is assigned to a human operator. That operator assesses the situation and chooses a level of autonomy to operate under: *goal-biased autonomy* or *teleoperation*.

In goal-biased autonomy, the operator helps the automation by providing desirable and/ or undesirable states for the robot to be in. The automation then makes use of this extra information to plan and execute an appropriate course of action (Côté, Bouzid, & Mouaddib, 2011). If the robot needs more

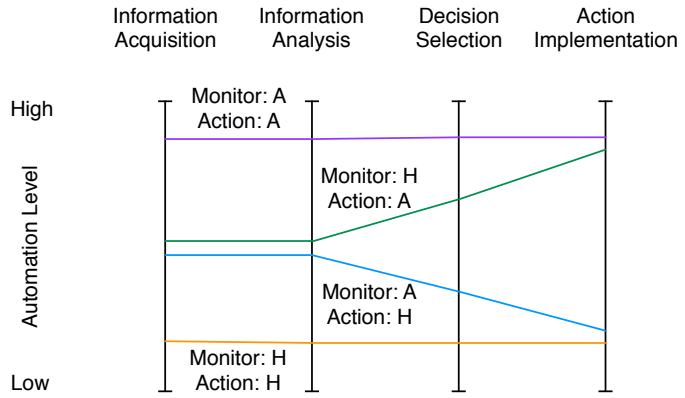


Figure 3.10: Characterisation of the Levels of Autonomy used by the system presented by Sellner, Heger, Hiatt, Simmons, and Singh (2006).

help, the operator can take over completely with the teleoperation level.

This system provides an interesting example of adjustable autonomy, as the responsibility for the level switching is shared between both the automation and the human operator (unlike other adjustable autonomy systems that tend to keep this switching solely human controlled or fully automated - see Section 3.4 for a more in depth look at this). The automation only has the authority to move out of the full autonomy level, and must defer to the human operator's opinion of which level to change to.

3.3.6 Sellner, Heger, Hiatt, Simmons, and Singh (2006)

Sellner et al. (2006) present a multi-agent robotic assembly system, where multiple robots assemble a structure together while a human operator supervises. The default state for the system to operate in is for all robots to act autonomously, and the human operator to only be called to act if the system deems one of the robots needs assistance. The system does not make use of traditional levels of autonomy as such, but rather splits the task into two components: i) action, the actual commanding of robot position etc. and ii) monitor, the detection of failures and task accomplishment. These two components can be assigned either to the automation or the human in any combination. These combinations form

Table 3.6: Levels of Autonomy and adjustment strategy for the system presented by Sellner, Heger, Hiatt, Simmons, and Singh (2006). A stands for automated control, H for human control.

Level Adjustment		**Levels of Autonomy**
Responsibility	**Trigger**	
Automated	Dynamic	Monitor:A - Action: A
		Monitor:H - Action: A
		Monitor:A - Action: H
		Monitor:H - Action: H

pseudo-levels which are characterised in Figure 3.10.

Changes in autonomy levels are made by the system, which monitors the actions of the robot and predicts its likelihood of success based on the time elapsed and the characteristics of prior successes or failures. Based on this prediction, the system can choose to change the level of autonomy, giving control to the human operator or letting the automation retry the task in case of a failure. It is worth noting that the system is capable of taking control away from the human operator using these same predictions (Sellner et al., 2006). (Martinez-Tenor & Fernandez-Madrigal, 2015) offer a similar system driven approach to adjustable autonomy, where the system can adjust its level smoothly and apply more or less automation to the operator's commands.

3.4 Analysis

The implementations showcased in the previous section provide examples for a range of levels of autonomy as well as a variety of strategies governing the changes between these levels. This variety is unsurprising if the motivations behind the use of adjustable autonomy in these systems is considered however. These systems can be broadly categorised into human-centred systems and automation-centred systems.

Human-centred systems, such as those presented by Goodrich et al. (2001), and Stentz et al. (2015), are mainly operated by the human operator and tend to use adjustable autonomy to relieve the operator of workload, allowing them some respite while the automation takes over. In

these systems, the responsibility for the system's level of autonomy lies with the human operator. Automation-centred systems, such as those presented by Côté et al. (2012), and Sellner et al. (2006), default to autonomous operation when possible, reverting to partial or total human control when the automation fails. In these systems, control over the level of automation is at least partially automated.

Another factor that stands out when comparing these systems' changes in levels of autonomy is whether or not these changes are planned (e.g. when doing this task the system is set to a particular mode), or dynamic (e.g. changes in levels which occur as a response to environmental changes). This distinction is not always clear cut, as a system with planned changes needs to be able to dynamically respond to errors for example, but the normal operating procedure for the system is likely to rely on one of these change strategies more than the other. Figure 3.11 organises the meta-control strategies discussed in the previous section into a grid. This diagram shows that that the automation-centred adjustable autonomy systems rely only on dynamic level changes, as the automation reacts to handle situations beyond its capabilities, while examples of human-centred adjustable autonomy systems that rely on all three triggers can be found. Systems that are built to undertake specific tasks will have their levels of autonomy designed with those tasks in mind, and therefore planned level changes, while systems built more generally will be designed with non-specialised levels of autonomy and react to their (potentially unknown) environment.

None of these systems come close to planned automatic changes in levels of autonomy. This however seems like a logical step forward for those systems that undertake known tasks. For example, the robots taking part in the DARPA Robotics Challenge such as CHIMP (Stentz et al., 2015), were designed to perform a specific set of challenges, and could practice performing these challenges ahead of time. During this practice, the level of autonomy best suited to each challenge would have been found and this level would have been used during the trials. Rather than having the operator select this best level during the trials, if the system could be made aware of the current task the robot is undertaking, this level could

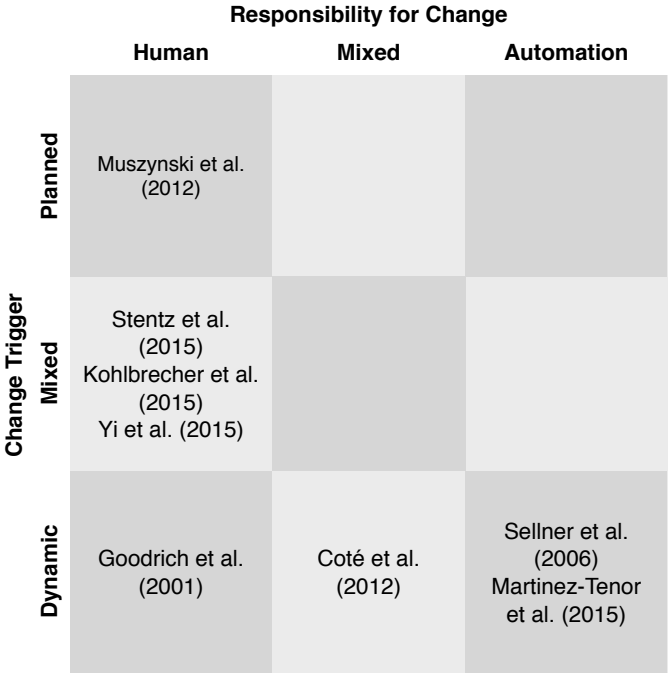


Figure 3.11: Adjustable Autonomy systems classified by their strategies for initiating change in their Level of Autonomy.

be set automatically.

3.5 Summary

This chapter discussed the concept of Adjustable Autonomy, the capability for a system to change the level of autonomy it employs, while that system is running. Adjustable Autonomy allows teleoperation systems to employ automation techniques in a more flexible way than they usually can be deployed in by providing the support of a human operator. Rather than having automation attempt to tackle all tasks facing the teleoperated robot, the level of autonomy of the system can be altered to allow the human operator to step in when needed. This has the additional effect of allowing the system to be run without or with lower operator input when able, allowing that operator some reprieve.

The mechanisms behind these changes in levels are complex, and have been classified in this thesis as being two dimensional. They depend on

the cause for the change in level, and the responsibility for the execution of that change. While most of the combinations of these two aspects have been explored by previous research, the combination of planned automated changes was found to not have been.

Section 3.2.2 presented the advantages that can be gained by enhancing teleoperation and autonomous systems with this capability, while Section 3.2.3 defined and provided examples for levels of autonomy, the discrete modes adjustable autonomy systems can switch between. Section 3.3 focused on the nature of these changes in levels, by providing a comparative analysis of five adjustable autonomy systems. Finally Section 3.4 presented an analysis of this information, and extrapolated on currently unexplored level change strategies, suggesting an avenue for further research.

Chapter 4

Assigned Responsibility: An Architecture for Mixed Control Robot Teleoperation

4.1 Overview

Teleoperation can be achieved through a variety of methods, ranging from full human control to fully automated control. Rather than select only one of these methods, some teleoperation systems choose several, and encourage active switching between methods to suit the situation at hand.

This brings along its own set of questions. Should this switching be manual or automated? Should it be planned ahead of time, or in response to environmental changes? What effects do mode changes have on operators' ability to control the remote machine?

The majority of systems capable of these changes choose the reactive approach, leaving the controller (human or automated) responsible for this task. This reactive approach is suitable for many applications as it is flexible and well suited to activities in unknown or unpredictable environments, but it represents yet another task for the operator already burdened with many cognitive and motor tasks. As discussed in Chapter 3, focus on this approach has left an unexplored gap in the spectrum of possible strategies for switching.

This thesis argues that in situations where the environment is known and/or predictable, pre-planning changes by explicitly assigning control methods to parts of the task could relieve robot operators of these decisions. In addition, such a strategy would provide a clear division of labour between the automation and the human operator(s) before the task even starts. Individual responsibilities being known ahead of time should limit operator confusion and could allow breaks to be planned for example.

The main contribution of this thesis (as stated in Section 1.6) is Assigned Responsibility: a new form of *adjustable autonomy*-based teleoperation that allows the selective inclusion of automated control elements at key stages of a robot operation plan's execution. Just like other shared control systems, the motivation behind this approach is to lessen the cognitive load, stress and fatigue on human teleoperators with the introduction of software aids. Assigned Responsibility strives to do this in a way that supports and encourages the gradual automation of the execution of the full task. To do this, Assigned Responsibility relies on pre-planning the execution of a task with a human manager or operator, assigning subsections of that task to either human or automated controllers before plan execution. This clear, pre-determined separation of roles is key to avoiding conflicts and confusions in an environment with several operators. The cost of this approach is some lack of dynamism in response to unforeseen problems.

In many professional settings, detailed advanced planning is a normal part of operations. The timing, costs, safety risks and scheduling requirements of such operations mean managers insist on this. While the approach taken by Assigned Responsibility might seem burdensome, it would likely be acceptable in these settings.

The remainder of this chapter is as follows. Section 4.2 presents and argues the case for Assigned Responsibility. Section 4.4 lays out the details of an architecture for Assigned Responsibility, while Section 4.5 details an architecture for Goal Accomplishment Tracking, an essential component of Assigned Responsibility. Section 4.6 considers the use of Assigned Responsibility in Learning by Demonstration, and finally Section 4.7 showcases a design for an Assigned Responsibility robot teleoperation

system.

4.2 The Theory of Assigned Responsibility

The previous chapters have outlined issues in the remote control of machines as well as the issues present in the solutions to this remote control. Regular one person, one machine teleoperation is very costly in terms of stress and fatigue on the human operator, and full automation is currently unable to provide the capabilities needed for many tasks in complex environments. Adjustable autonomy systems could strike an effective balance between capability and operator load, providing a promising avenue for research in robot teleoperation. Chapter 3 described some mechanisms and strategies used by adjustable autonomy systems to effect their changes in autonomy, based on the responsibility (human or automated) and the trigger (planned or dynamic) for those changes. A gap in those mechanisms and strategies was identified, located around planned automated changes.

To explore this gap, this thesis proposes Assigned Responsibility: a framework allowing for the remote control of one or more machines by one or more operators, human or otherwise. More specifically, Assigned Responsibility can be described as a shell for a pre-planned adjustable autonomy teleoperation interface. It allows the level of autonomy of a particular robot to change systematically during the execution of an overall plan. Unlike other adjustable autonomy interfaces, these changes are pre-planned and are triggered automatically at specific points of a plan's execution. Figure 4.1 shows where Assigned Responsibility sits relative to other assigned responsibility systems. Assigned Responsibility requires a job to be broken down in to steps before its accomplishment, with each of these steps assigned a level of autonomy. During execution, the system then keeps track of progress through the work, adjusting the level of autonomy as required when steps are accomplished.

By having level changes pre-planned and switched at step boundaries, Assigned Responsibility necessarily sacrifices some of the flexibility that has made adjustable autonomy interfaces successful. This tighter control over

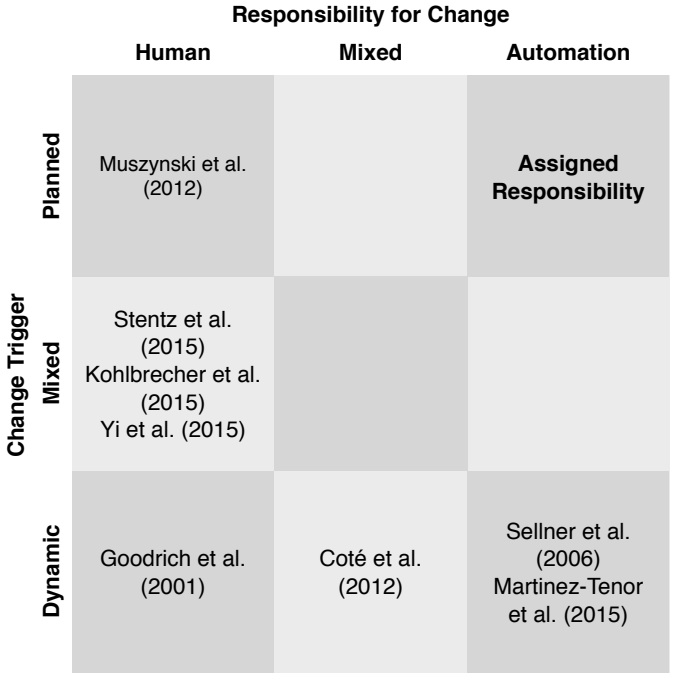


Figure 4.1: Levels of autonomy changes in Assigned Responsibility and other adjustable autonomy systems.

changes in levels of autonomy is motivated by four independent factors: i) A predictable schedule of responsibility allows human operators to know ahead of time when they are and when they are not needed, allowing them to plan breaks. ii) The pre-set levels of autonomy should reduce conflicts between operators, as their role at each stage is known to them ahead of time, and clearly signalled at each step. iii) Pre-determined changes provide clearcut boundaries for automation to operate in, reducing programming complexity, and allowing for the gradual automation of the whole task. iv) In scenarios where the use of automation is restricted due to laws, ethical considerations, or simple mistrust in the capabilities of that automation, the ability to specify what is and what isn't automated becomes necessary.

It is worth noting that Assigned Responsibility as proposed here is not aimed at superseding other adjustable autonomy approaches, but rather that the currently unexplored combination of planned automated changes in autonomy is capable of outperforming traditional control approaches. Assigned Responsibility is not suitable for all deployment scenarios,

because pre-planning entire tasks is not always possible in fast-changing environments, where a more dynamic approach would be more appropriate. In scenarios where pre-planning is possible, or mandatory, leveraging this possibility should be an advantage.

To function, an Assigned Responsibility system requires many of the same basic elements as any adjustable autonomy system, namely an interface for the human operator, some control automation, and the means to handle varying inputs from both. However, it also needs a management module, responsible for automatically keeping track of the current progress through the task so as to enact changes in autonomy when specified. More specifically, an Assigned Responsibility system needs to satisfy the requirements presented in Table 4.1. The table presents these requirements in terms of the broad area of a system they cover, the capability required in that area, as well as how to validate the system to check whether the requirement is fulfilled. These requirements are specifically aimed at a control system for a single robot (in this case the system described in later chapters), and could be altered to fit other use cases.

4.3 Assigned Responsibility in Use: An Example Scenario

This section presents an example scenario of a human operator controlling a robot through a task using an Assigned Responsibility system. The intention if this section is to translate the theoretical information presented in the previous sections into a look at what the practicalities of using an Assigned Responsibilities could be.

In this scenario, the task set to the operator/robot team is a DARPA Robotics Challenge-like task of controlling a robot through a series of challenges: i) drive the robot up to a door with a sign on it, ii) open the door, iii) drive through the door until facing a target position marked on the wall. The robot in this scenario is a wheeled robot equipped with an arm mounted manipulator long enough to reach the door handle.

This challenge being known ahead of time, this task was broken down into a series of goals for each of which a level of autonomy was set from the following list of available levels: i) full human control (no automated assistance), ii) high-level human control (human sets high level commands, automation then executes), iii) full automation (no human input). The following list displays the responsibility of goals are assigned as well as a description of the accomplishment of the goals:

1. **Face the door: full human control.** The operator uses the controller to guide the robot towards the door, using the onboard camera to ensure the robot is somewhat normal to the door and that the sign on the door is clearly visible in the video feed. The accomplishment tracking software analyses this feed throughout the task. Once the sign is recognised by software, the goal is deemed accomplished and the system moves on to the next goal.

2. **Grasp the door handle: high-level human control.** The operator clicks on the video feed to identify the location of the door handle. Using that information as well as the distance between the robot and the door reported by the range sensors mounted on the robot, the automation calculates the coordinates of the door handle and reaches out with the manipulator to grasp the handle. Minor positional adjustments are made to the arm as the handle approaches using the arm mounted camera's feed. The goal is deemed satisfied when the manipulator's force sensors report that the manipulator is applying pressure (and is not fully closed).

3. **Turn handle and open door: full automation.** The operator is notified of the previous goal's accomplishment as well as the fact that the system is now in full automation mode: human inputs are no longer accepted. The automation takes the robot through a carefully planned set of motions that allow the door to be pushed open, the arm rotating to account for the door's swing as the robot advances. The accomplishment tracking system registers the fact the door is open once the door is only visible on the arm camera's video feed, the front facing camera has now lost all sight of it.

Table 4.1: Requirements for Assigned Responsibility Systems

Area	Required Capability	Requirement Validation
Human Interface	The system must provide a usable robot control interface	Show usability through a usability evaluation of the interface
	Must support changes in levels of autonomy by dynamically changing to suit the current mode the system is operating under	Prove interface changes appropriately during trial runs
	Must support shared control modes by providing methods for the automation and human operator to interact	Prove interface allows human-automation cooperation during trial runs
	Must provide goal context to operators	Show human operators have access to a plan progress display
Automation	Must be capable of accomplishing assigned goals	Prove automation capabilities during trial runs
	Must support changes in levels of autonomy	Prove automation changes according to current level during trial runs
Level and plan management	Allow pre-setting of goal/level pairs	Prove the system supports this process
	Accurately recognise goal accomplishment	Show that goal tests successfully identify goal accomplishment
	Communicate changes in goal states to the rest of the system	Show that when goal states change, the rest of system responds
Robot	Possess the capabilities required by the tasks	Show the robot can perform in trial runs
	Provide appropriate sensors for operation and progress tracking	Prove the sensors on the robot are appropriate in trial runs

4. **Drive through: full human control.** The operator resets the arm's position to default using a button on the controller, then drives the robot forwards until facing the position marked on the wall. The accomplishment tracking system recognises the target visually and marks the goal as accomplished.

4.4 An Architecture for Assigned Responsibility

This section proposes an architecture supporting Assigned Responsibility in robot teleoperation systems. This architecture is general, and not aimed at supporting any particular robot teleoperation systems. It is rather intended as a retro-fit onto existing robot teleoperation systems, in that it does not intercept the operator/robot control loop, but rather sits on top of that basic architecture, communicating with the operator's station, and observing the robot. This base system should possess the ability to operate in several different levels of autonomy to make full use of Assigned Responsibility. The core of the proposed architecture focuses on the *management module*, a self-contained block in charge of ensuring the level of autonomy of the system is always set to the level specified before the task started. This is performed through the operations of two main processes, *accomplishment monitoring*, and *plan management*.

The remainder of this section introduces and defines the concepts of plans and goals as used by the architecture, and then goes on to describes the management processes in detail as well as their relationship and role in the greater system.

4.4.1 Plans as Trees, and Goal Breakdown

To enable Assigned Responsibility, tasks must be segmented. This architecture proposes to do this by decomposing jobs into a series of goals, called *the plan*. In many domains (e.g. automated planning), a plan is represented as a graph tree in which the nodes are the goals (the individual steps), the edges between them are actions, and goals are defined as desirable states of the world (Ghallab, Nau, & Traverso, 2004). Achieving a goal amounts to choosing and then performing actions that are required

to change the state of the world from an undesirable state to the desirable state specified by the goal. The plan is a tree of interdependent goals, with each parent goal relying on the fulfilment of all its child nodes. The job is accomplished when the root node (the highest-level goal) is satisfied.

These tree representations are useful for Assigned Responsibility, as they are easily describable using graph theory, which provides useful rules to read and understand these graphs automatically. The plan tree graph also has the advantage of being very human-readable, trees being a natural way for people to break down tasks (Sacerdoti, 1977). This allows the human operator to be very clear about the plan.

To allocate tasks to either the human operator or automated control system, it is necessary to break down the top level goal into sub-goals, and those sub-goals into further sub-goals until a satisfactory goal granularity is obtained[1]. The resulting plan graph possesses the properties of an ordered rooted tree, the details of which are described below, and illustrated in Figure 4.2.

- It can be defined as a connected acyclic graph G with $G = (V, E)$ where V is a collection of n vertices and E is the collection of $n - 1$ edges.
- The vertex (V_1) is designated as the root of G.
- The parent of a vertex is the vertex connected to it on the path to the root; every vertex except the root has a unique parent.
- A child of a vertex V_n is a vertex of which V_n is the parent
- An ordering is specified for the children of each vertex.

[1]This is task and implementation specific, but mostly means a self-contained piece of work that can be accomplishable using a single process. The goal of closing a door could be broken down into two subgoals, navigating to the door and closing the door for example. It could also be broken down into more subgoals, if required by the available automation. At some stage however, it is likely that overly small subgoals will frustrate human operators, who will resent having to accomplish baby steps. A balance between the needs of automation and the wants of human operators will have to be struck, possibly on a task to task basis. Finding this balance will be the role of the person or automation in charge of plan creation.

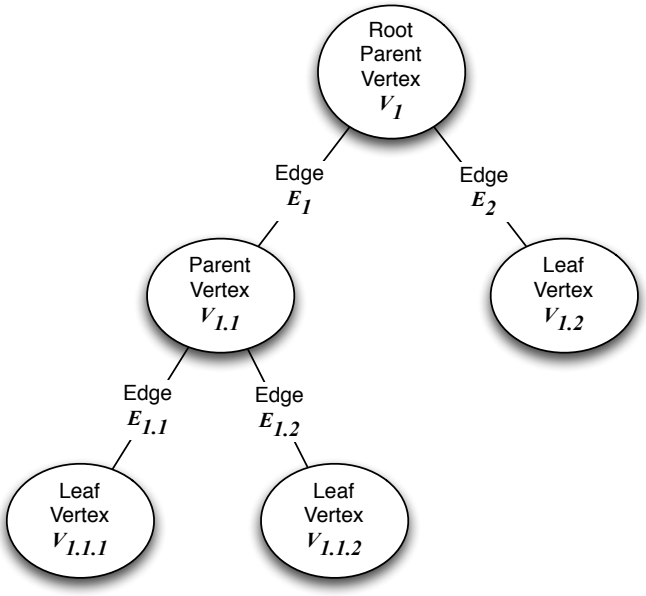


Figure 4.2: Sample ordered rooted tree. This diagram illustrates the different components a plan graph can possess: root, parent and leaf vertices, all connected by edges. The vertices are ordered according to their numbering.

- A terminal vertex (or leaf) of a tree is a vertex of degree 1.

We can interpret these theoretical concepts as follows:

- The root is the overall goal to achieve, each other vertex is a sub-goal.
- To achieve a particular sub-goal, its children have to be satisfied, in order (By convention, left to right).
- By extension, once all sub-goals are satisfied, the root goal is also satisfied.
- The leaves are the lowest-level goals, and are the only goals satisfiable directly. Satisfying all of the leaf goals, in order, satisfies the root goal.

4.4.2 Accomplishment Monitoring

To allow automated changes in levels of autonomy to happen as planned, an Assigned Responsibility system must be able to track progress through

the execution of a task to completion. Current research interest in the monitoring of the execution of plans is focused on supporting the improvement of this execution, and the improvement of the monitoring itself (Lee et al., 2009). These improvements are initially motivated by the specific domains they originate from, but tend to be generalised later if the ideas are not domain-specific. For example, in the business domain, languages such as BPEL (Business Process Execution Language), seek to describe the execution of business processes in order to provide some process management capabilities, such as lifecycle management, failure recovery, and a variety of control regimes, while mostly ignoring the data being transferred between those processes (Leymann, Roller, & Thatte, n.d.). Scientific workflow processing, such as is supported by workflow management engines (Deelman et al., 2004) is driven by the need to operate on large data sets (Sonntag, Karastoyanova, & Deelman, 2010), therefore placing more importance on the data flowing between processes. Regardless of low-level differences, these systems provide similar high-level plan execution monitoring functionality, and seek to solve similar problems at that level.

To track the progress of a plan's execution, a monitoring system needs to be able to actively fetch or wait for an update on goal statuses during execution. BPEL does this through web service interfaces, whilst scientific workflow management systems commonly utilise log file parsing. After the data is collected, it needs to be analysed for patterns, simple and complex, that indicate the current status of plan execution. Depending on the state of the plan execution, it can continue, or be re-planned. This process has been formalised by the Autonomic Computing community to support adaptive systems (Kephart & Chess, 2003).

The segmentation of tasks into plans of goals brings along advantages for accomplishment monitoring in Assigned Responsibility systems. It is convenient to represent goals in the same form as observed states of the world as provided by sensors. Checking the accomplishment of a goal might then be as simple as comparing it with a current set of world-state representations. For example, if a robot is equipped with a sensor which returns its position in two-dimensional space, a locational goal might be

specified as 'robot_at(250, 300)'. This can be directly compared to the output of the sensor, which might return 'robot_at(240, 300)'. A more sophisticated arrangement would involve abstracting the goal to 'robot_at(waypoint)' and providing additional knowledge defining the waypoint in terms of a number of sensory tests and satisfaction conditions with respect to those tests. In this case, the waypoint is associated with a set of GPS coordinates of an area within which the target is found, a specific barcode known to be present at the target, or an image pattern to be matched against a known landmark through the use of an on-board camera. The satisfaction condition might be that the current GPS coordinates must be within bounds, and that a match on either of the other sensory indicators would be sufficient. Such straightforward arrangements would not always be enough, because the relationship between sensory data and actual world states is not always simple. Not only does the sensitivity, range and signal-to-noise ratio characteristics of a given sensor affect the interpretation of its signals, but the satisfaction or failure of some goals might involve a subtle alteration in sensed properties, possibly including necessary state progressions or alternatives. Some of the literature on robot perception deals with the control of uncertainty introduced by these complications (Thrun, 2000) (Minguez & Montano, 2005).

Not all goals are the same, but may have different natures based on their objective and relation to processing. The proposed architecture supports three types of goals:

- **Achieve goals** (Dastani, Riemsdijk, & Meyer, 2006) are simple expressions denoting a desired world state to be reached.
- **Maintain goals** (Dastani et al., 2006) are goals that need to be protected (i.e. their accomplishment tests must not be allowed to fail). Maintenance goals require extra monitoring after they have been initially accomplished, placing an extra burden on the monitoring system.
- **Opportunistic goals** are goals associated with a watch for particular events or world-states, the presence of which is considered favourable.

> Opportunistic goals mirror maintain goals, in that rather than demand checks for threats to goals, they encourage checks for contingencies favourable to goal accomplishment.

These goal types require different monitoring support. Achieve goals depend upon matching the required world-state to the current state of the world. Maintain goals seek to actively protect a desired state. This requires tests sensitive to boundary conditions around the goal state which suggest a threat, requiring protective actions to be executed. These safeguards can be included the hierarchical plan graph as special annotations. Opportunistic goals must use tests to detect occurrences known to promote the accomplishment of goals, as well as the appropriate actions (such as skipping ahead) which may be similarly included in the plan.

4.4.3 Plan Management

The plan management module is charged with ensuring all of the components of the Assigned Responsibility system are aware of the current state of the task's achievement. This state record is used to synchronise all of these components, ensuring the level of autonomy of the system is as planned, the human operator is aware of the current goal, and that the accomplishment monitoring process is checking for the accomplishment of the correct goal. The plan manager relies on the accomplishment monitoring process for updates to the task progress, and is charged with integrating and disseminating those updates to trigger appropriate changes to the human-machine system.

4.4.4 Example Responsibility Assignment Strategy

Being a type of adjustable autonomy, Assigned Responsibility needs to handle levels of autonomy. By applying the general theories put forward by (Miller & Parasuraman, 2003; Parasuraman & Miller, 2006; Miller & Parasuraman, 2007) to the domain of teleoperation, and Sheridan and Verplanck's (Sheridan & Verplanck, 1978) automation scale, a set of

Table 4.2: Levels of Autonomy for Assigned Responsibility

Mode	Full Name	Description
H	Full Teleoperation	The human operator controls every aspect of the teleoperation process.
H/A	Assisted Teleoperation	The human operator controls the teleoperation process, but is assisted by simple automated systems. For example inverse kinematic control of a robotic arm.
A/H	Human Assisted Automation	The automatic controller executes low-level robot controls. The human operator provides high-level help such as designating waypoints in navigation tasks, and object location in manipulation tasks.
A	Full Automation	The automatic controller has full control.

generic *levels of autonomy* for use with Assigned Responsibility was created, as described in Table 4.2.

In Assigned Responsibility each of the plan's sub-goals are explicitly assigned one of these levels before the execution of the plan. In this example assignment strategy, it is assumed that the human operator is capable of effectively controlling the robot and satisfying all of the goals using it, and that the preference is to relinquish most of the control to the automated modes in order to free up the human operator. These assumptions are for explanatory purposes only, and should not be taken for granted. An example assignment process is described below. This process might not be done by the operator, rather by a manager for example (which could be human or automated), in which case making the operator aware of the contents of the plan is a critical step.

1. All of the goals are set to full teleoperation.

2. Each leaf goal known to be at least partially automatable is set to the appropriate level[2]. As the automatic software is programmed to

[2]Like many of the details of this architecture, this could be determined in an automated manner or left to a human manager. In either case, the goals in the plan have to be matched up to the capabilities of available automation modules. This could be simply done using

accomplish more goals, more and more of the leaf goals will be assigned automated levels.

3. The resulting list of goal and associated assignments is shown to the human operator (if needed).

4. The final assignment is set, ensuring both the human operator and the automated control system are aware of the goals for which they are responsible.

4.5 An Architecture for Goal Accomplishment Tracking

This section proposes an architecture to provide goal accomplishment tracking to support accomplishment monitoring as described in Section 4.4.2. This architecture is designed to remove the need for agents (human or automated operators, referred to as agents here for simplicity) to report progress at every step, shifting this role back onto an analysis module in charge of the tracking of plan execution progress. This module monitors the world, using data provided by sensors, for the changes expected to occur when each goal has been accomplished. The module then updates a central plan-tracking structure with the progress made (See Figure 4.4).

This approach is centralised by design, to account for the varying range of capabilities displayed by control agents in various domains. While some automated agents, such as those controlling robots, might perform goal accomplishment tracking internally to ensure proper execution of their assigned tasks, other more basic agents simply encounter a situation and act in response without ever checking to see if their goal is accomplished. In the case of human or human-controlled agents, goal accomplishment is understood by the human in question but this is not easy to broadcast back to other agents without additional tools. A centralised approach to goal accomplishment tracking helps mitigate the issues caused by these

a lookup table, but more interesting possibilities could be considered. For example, if statistics of task success rates are kept, the system could learn from previous attempts to determine the most suitable level of automation.

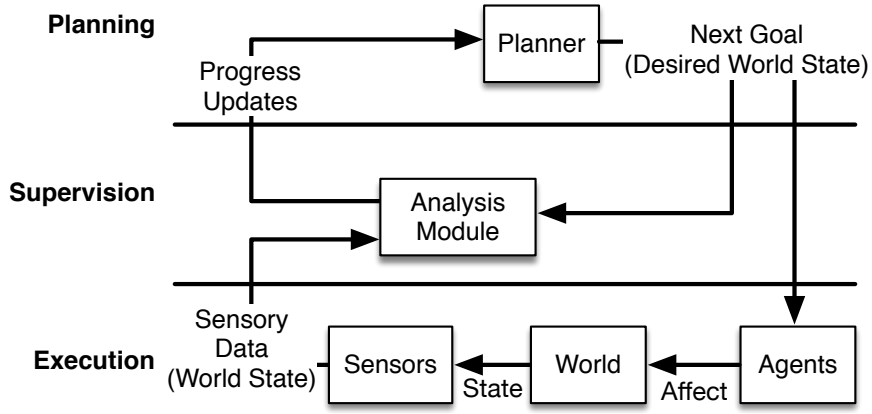


Figure 4.3: Architecture for Assigned Responsibility.

differences by making no assumptions about the communications capabilities of the agents being supervised. Note that because it is only world states that are generally used to decide on accomplishment, it does not matter wether the robot itself or some other agency has satisfied the goal.

4.5.1 Pre-requisites

The architecture presupposes the presence of several elements (See Figure 4.3), without which it has neither the means nor the reason to fulfil its purpose.

- **A plan** to provide both a reason for the architecture to be in place, and the framework for the analysis module to operate within. Plans allow agents to know what changes to make to the world state, and inform the analysis module of what changes to expect.

- **One or more agents** to effect changes on the world. Agents accomplish the plan's goals, creating changes that can be tracked.

- **Sensors** to allow the reading in of the state of the world. They allow changes caused by the agents to be observed and tracked.

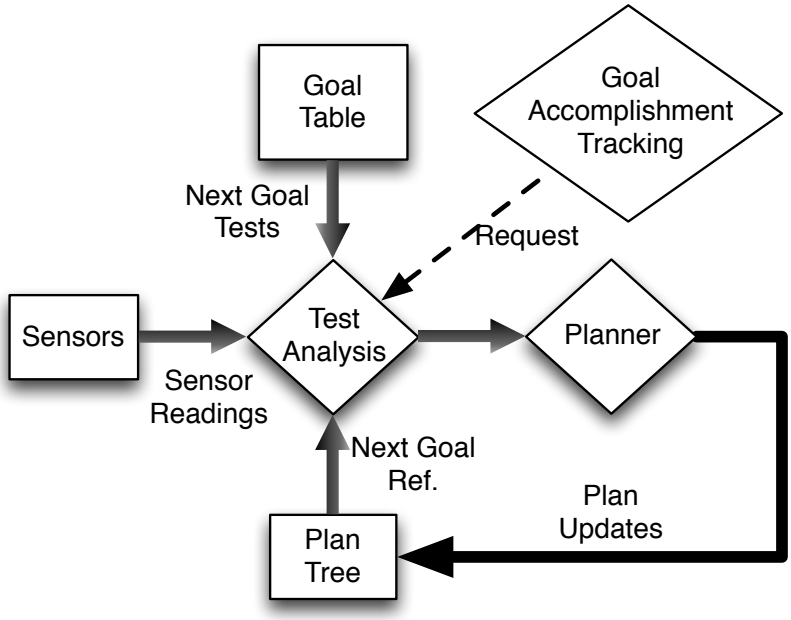


Figure 4.4: The supervision process: data flow.

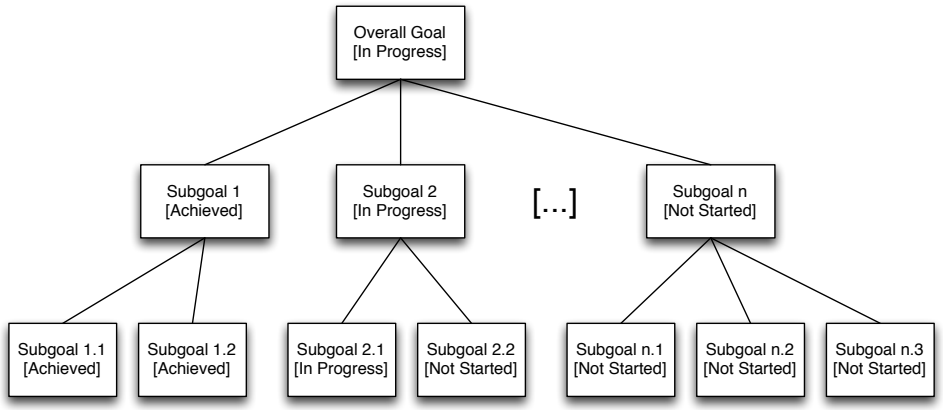


Figure 4.5: Representation of a plan structure supporting Goal Accomplishment Tracking. Plans show the relationship between goals, as well as contain progress information.

4.5.2 Runtime Activity

During runtime, the proposed architecture feeds sensor inputs through the analysis module, to perform the tests appropriate to the current goal. Figure 4.4 shows the way data progresses through the architecture. Sensor information such as GPS coordinates are fed in by the appropriate sensor, to be compared against a target coordinate provided by the goal table. The correct test is extracted from that table using the current goal provided by the plan.

a. Inputs

For its most basic functions, the analysis module requires three types of inputs: the plan's structure, desirable world states and actual world states.

- **The Plan Tree** is the data structure that holds the logical information about the goals. This includes the overall hierarchy and the relationship of the goals to one another, and goal status (i.e. achieved, in progress, not started, and maintaining). A representation of an appropriate plan structure is presented in Figure 4.5.

- **Actual World States** consist of the information reported by the sensors queried by the architecture. This takes a wide variety of forms as all sensors tend to return their own form of data. This means that to be usable, this information must be made understandable to the system. This could require conversion from the sensor's format to an compatible representation, or the addition of a parser for the data in the system itself.

- **Desirable World States** are generated during the preparation step (See Section 4.5.4). These states are fetched by the supervisor from a data storage element, such as a lookup table. They must follow the same format as the actual world states to enable comparisons.

```
for each goal G_i do
    flag ← TRUE
    for each sensor S_k in satisfaction_conditions (G_i) do
        if needed(S_k, G_i) then
            if power_up(S_k) then
                wait(max.sensor.powerup.time)
            end if
            if inactive(S_k) then
                message: "Can't test G_i, sensor S_k is inactive"
                flag ← FALSE
                break
            else
                if ¬ match(S_k[W_l], S_k[W'_l]) then
                    flag ← FALSE
                    break
                end if
            end if
        end if
    end for
    if flag ≡ TRUE then
        message: "Goal G_i is satisfied"
        leave_on(G_{i+1})
    end if
end for
```

Figure 4.6: Example algorithm testing for goal satisfaction, including sensor power management.

b. Analysis

Processing the sensor data during runtime can be accomplished using a relatively simple algorithm. An example generic solution is presented in Figure 4.6. This algorithm iterates through the goal table, checking for each goal (i) the availability of the required sensors, and (ii) whether or not the sensor data matches the desired results. The algorithm keeps looping through the tests required for each goal until these tests are satisfied. Once the tests are satisfied, the goal is set to accomplished and the algorithm moves on to testing the next goal, until all the goals have been satisfied.

The algorithm calls on several functions not included in Figure 4.6 for brevity. They are explained below:

- **needed(S_k, G_i)** returns false if i) the satisfaction conditions of G_i

contain an expression in disjunction with S_k, and ii) that expression currently evaluates to true; true otherwise.

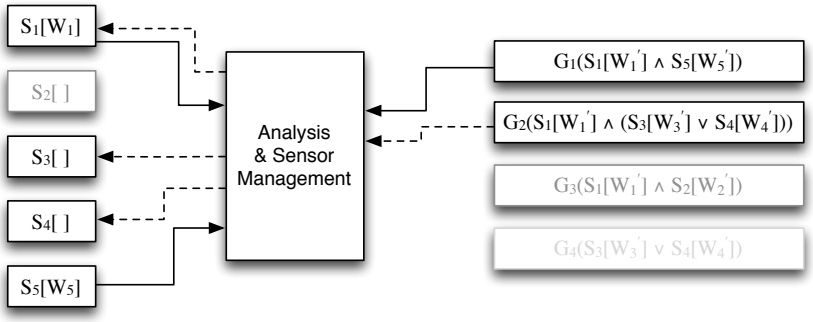
- **power_up(S_k)** checks to see if S_k needs powering up. If so, powers up S_k and returns true; false otherwise.
- **inactive(S_k)** returns true if sensor S_k is not currently returning a valid world observation; false otherwise.
- **match($S[W], S[W']$)** returns true if world observation W agrees with corresponding goal observation W'; false otherwise.
- **leave_on(G_{i+1})** checks if the next goal (G_{i+1}) requires any sensors already used by G_1; all others are powered down.

Note that this version of the algorithm does not distinguish between the three types of goals. To do that, the algorithm of Figure 4.6 would need to be modified to provide special processing for maintenance goals and for opportunistic goals. Maintenance goals would need extra guard tests and a stack of monitoring processes. Opportunistic goals need equivalent extra tests for favourable circumstances with the associated stack of monitoring processes. Once a goal was processed, these processes would run continually until the algorithm terminated.

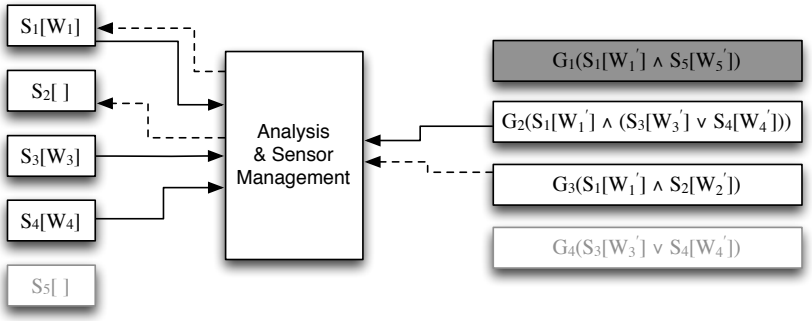
c. Output

To share the results of this monitoring, the architecture needs to be able to send messages. In its most basic form, the architecture will post progress updates on the plan data structure. This structure is centralised to allow updates to be rapidly propagated to each other elements of the system; for example, agents can periodically query this central structure to get an up-to-date picture of the plan's completion status. This information can be then used to inform the human operator of the current state of the plan, and ensure the correct level of autonomy is being used for example.

The number of outputs will vary based on individual implementations and their needs for extra functionality. For example, a system requiring fault tolerance (See section 4.5.3), will need to be able to request a form of re-planning.



(a) Accomplishment tracking of goal G1, and sensor readying for goal G2.



(b) Accomplishment tracking of goal G2, and sensor readying for goal G3.

Figure 4.7: Goal Accomplishment Tracking and Sensor Management. G_n are goals, S_n sensors, W_n world states, and W_n' desired world states. The left section of the diagrams list available sensors, their state (greyed out means off.), and if they are reporting world states. The right section shows the information contained in the goal table. Which sensors the goal test relies upon, as well as the desired world states.

d. Sensor Management

Managing the fusion of data from multiple sensors to test for a single condition, and dealing with the number of different sensors required by the architecture to be useful, both create issues that must be overcome. The sensor fusion problem (properly analysing and extrapolating information about the real world using data from multiple sensors, even when they may be reporting contradictory readings) needs to be considered for each goal if the accomplishment tracking for that goal relies on the input of multiple sensors. One simple solution is to express the relationship between the sensors as a set of Boolean functions as shown in Figure 4.7.

The specification of how sensory tests apply to each goal is made during the preparation step (see Section 4.5.4).

The advantage of knowing which sensors will be needed for each goal is that predictions about sensor uses can be made. This enables us to: (i) enact power management policies (e.g. sensors can be turned on only when needed, and off when not), (ii) ensure sensors are turned on early to ensure any sensor startup delays are accounted for, and (iii) check sensors for malfunctions ahead of use, leaving time for any potential re-planning to be made. This sensor management is illustrated in Figure 4.7, with Figure 4.7a showing a possible state for the system to be in when supervising the accomplishment of a goal G_1. G_1 requires tests on data from sensors S_1 and S_5, providing world states W_1 and W_5 respectively. These will be compared to desired world states W_1' and W_5'. The system is also pre-emptively preparing sensors S_1, S_3, and S_4, which are required by goal G_2. Figure 4.7b shows the state of the system after goal G_1 is accomplished. The accomplishment of goal G_2 is being supervised while sensors required by the monitoring of goal G_3 are being prepared. This kind of management is especially important for mobile robots, as it limits the use of power and computing resources.

4.5.3 Fault Tolerance and Optimisation

A fault tolerant system has the ability to respond to any event that might cause a problem at runtime (Randell, 1975). The aim of execution optimisation is to take advantage of opportunities to improve the execution of plans during runtime (Lee et al., 2009). Both fault tolerance and optimisation are features of many current workflow monitoring systems. Proposing approaches to dealing with these processes is beyond the scope of this thesis; however, facilitating the inclusion of such capabilities in this architecture will broaden its usefulness. Fault tolerance and optimisation are existing concepts that are applied in many planning scenarios, and Assigned Responsibility's reliance on plans means a logical next step would be the inclusion of such capabilities.

Both fault-tolerance and optimisation processes require some monitoring of the system, making the proposed architecture well suited to

trigger and inform those processes. Any condition requiring fault-tolerance or optimising processes to engage could be specified as a maintenance (or opportunistic) goal in the overall hierarchy, the realisation of which would trigger those processes. Figure 4.4 shows how any re-planning needed by these processes could be kept distinct from the analysis module, only having an impact on the inputs and outputs used by the analysis.

4.5.4 Goal Accomplishment Tracking Preparation

The architecture proposed here is applicable to a wide range of scenarios, and as such is designed in a high-level and modular fashion. To prepare the automatic goal accomplishment tracking for a particular application(industrial maintenance by robot), the following steps have to be performed. This process can be performed by humans, and written to files accessible by the system, or automatically if the capability exists.

1. **Enumerate available sensors**. Sensors available to the architecture must be found and recorded in a list. It is necessary to include in the list whether or not the sensors are bound to a particular location or are mounted on a mobile platform.

2. **Extract goal list**. Before a series of tests for each goal in the plan can be made, a list of those goals must be extracted from the plan.

3. **Match goals with sensors**. The two lists must be cross-referenced to associate each goal with at least one sensor. This match up will depend on several factors: (i) the ability of the sensor to produce meaningful information about the goal's status, and (ii) the location and mobility of the sensor (i.e. can the sensor 'sense' at the right location. For example, a camera pointing away from an object would be unable to provide information on a robot's proximity to that object).

4. **Convert abstract goals to usable world states**. Each goal must be converted from an abstract meaning to usable boundaries with which sensor data can be classified. This forms our desirable world states. The form these states take can vary, image recognition tests

would use an image as a world state, whereas a location tests would store coordinates.

This process is demonstrated through an experiment in the next chapter, where the architecture is tested with real data collected by mobile robots sensors in the field.

4.6 Opportunities for Learning by Demonstration

The teleoperation experiments in our laboratory require examples of automated controllers to be mixed with human control. One of the benefits of teleoperation is the opportunity to learn skilled actions from a human expert, delivered directly to the machine in real contexts (Mann & Small, 2012). Learning by Demonstration (LbD) is one approach, involving the extraction of a robot control policy from a set of demonstrations of the target skill, recorded from the sensors and motor outputs (for a review see (Argall et al., 2009)). While this topic is not a focus of the thesis, it is worth noting that the Assigned Responsibility approach could both support and benefit from LbD.

Goal accomplishment tracking is important in the LbD context because it can be used to partition large demonstration datasets containing multiple, possibly very different functional relationships into manageable, more easily learned sub-units. Goals represent an important constraint in high-dimension, noisy learning problems. For example, Isaac and Sammut (2003) were able to use human demonstrations of basic flying manoeuvres to learn compact, robust and transparent controllers ('behavioural clones') which both discovered how the pilots were setting goals and then learned how to cause the simulated aircraft to meet those goals. By learning how to set goals, and then learning how to control a dynamic system to reach those goals, the cloned behaviour can be learned more easily and show much greater robustness to changed conditions or plans than when learned from undifferentiated demonstrations of an entire task. Automatic goal monitoring allows switching between these learned modular units of behaviour during the execution of complex tasks in dynamic environments, whether by human or robotic agency.

Figure 4.8: The commercial Coroware robot modified for this project approaching a work site for inspection.

4.7 A Modular Design for the Assigned Responsibility-based Control of a Mobile Robot

4.7.1 Overview

Automatic maintenance of physical equipment requires a mobile robot to periodically visit a number of worksites. The robot may perform tasks such as photography, gathering sensor data on environmental conditions, physically probe the integrity of surfaces, joints or attachments, remove panels and/or to change out faulty components (Mann, 2008). To perform such tasks, a robot needs to be guided around multiple worksites (as shown in Figure 4.8), aligning itself close to each one in turn so as to be able to address important objects.

This section describes a modular design for an Assigned Responsibility-based control system for a small mobile robot, allowing that robot to perform the task described above under human and automated supervision.

4.7.2 A Modular Approach

In the maintenance scenario described above, and in most robotics applications, a robotic control system must operate over a network; at the minimum the robot itself and a control station. A modular or component-based approach is ideal when designing software that will be distributed over multiple hardware platforms connected over a network.

In a modular design, individual components should be interchangeable with other components capable of performing similar tasks. This is important in robotics, where a system might want to control another robot, or use a different planning algorithm, or in the case of goal accomplishment tracking, load up another goal test.

For an Assigned Responsibility system, the most important reason to choose a modular approach is the drive to progressively automate the execution of tasks. As automation for more tasks is written, it will be added to the overall system. The easiest way to allow this to happen is to make the automated controller modular in nature: when new automation techniques become available, modules implementing them can be created. Adding new automation capabilities to the system then simply becomes a case of making these new modules available to be loaded up during runtime. One of the advantages of having humans controlling the robot during non-automated tasks is that data can be collected on the human accomplishment of these tasks, possibly paving the way for future automation.

4.7.3 Design for an Assigned Responsibility Teleoperation System

This design consists of components divided into three groups: Operation, Execution, and Management (As shown in Figure 4.9). The Operation and Execution groups include the basic teleoperation loop of operator sending commands to the robot and the robot executing them as well as sending feedback to the operator. The management group is concerned with tasks specific to Assigned Responsibility. These groups are described in detail below.

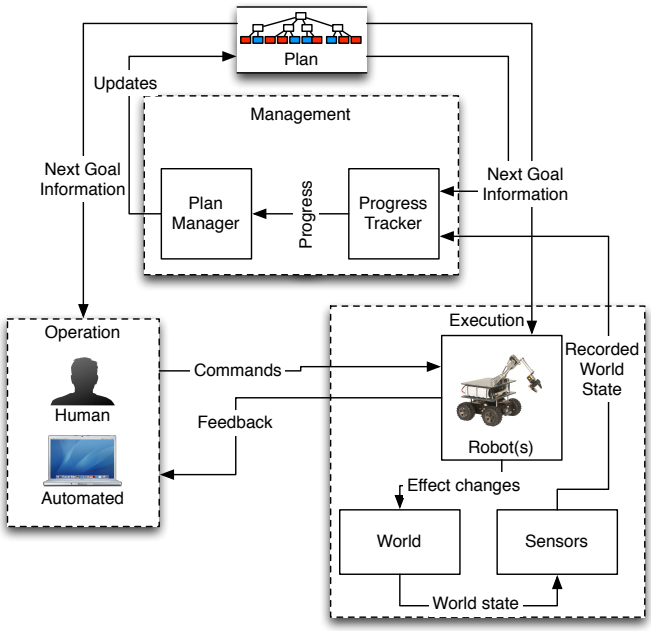


Figure 4.9: Proposed design for an Assigned Responsibility system.

a. Operation

The operation group provides an abstraction layer for the command side of the remote control. Any changes in operators, or use of multiple operators should be handled within that group. The operation group contains the following modules: the user interface for the system, the automation modules, and the modules handling mixed input modes (when human and automated operators control the robot together).

b. Execution

The execution group is mostly concerned with executing commands sent by the operation group, and with handling sensors mounted on the robot. It includes the main control loop for the robot, as well as an interface to handle requests for information gathered by the robot (mostly sensor data).

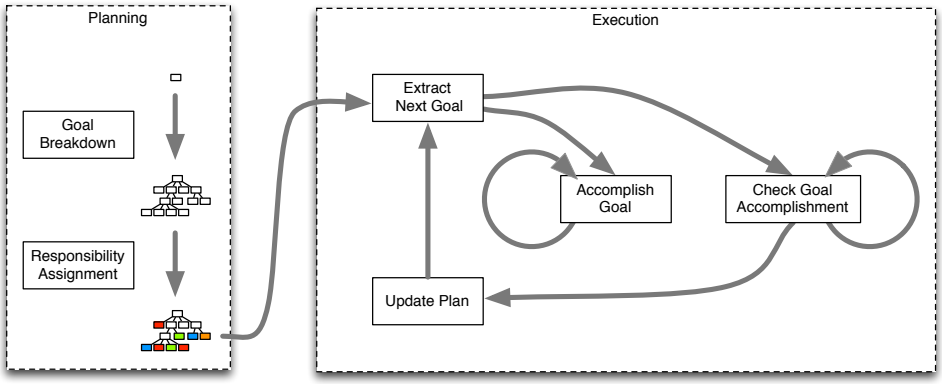


Figure 4.10: Example task-execution timeline for an Assigned Responsibility system. The transitions between states are triggered by the accomplishment of those states, as determined by the specified test conditions.

c. Management

The management group is in charge of two major tasks: plan management, and progress tracking. The plan management part is tasked with managing the file storing the plan, making the plan and its goals available to modules who request it, and updating the plan with new information (e.g. goal accomplishment). The progress tracker monitors the plan's execution in the real world using sensor data to keep the plan up to date.

d. Example Task-execution Timeline

A task's accomplishment can be broken down into two stages: planning and execution, as shown in Figure 4.10. The planning stage starts with the break down of the job into goals and sub-goals. The goal tests required for the accomplishment testing of these goals are determined. Finally responsibility to satisfy each sub-goal is assigned a level of autonomy. This design makes use of the levels of autonomy described in Table 4.2. The execution stage requires the system to loop through four stages for each of the goals identified in the planning stage: i) Extracting the goal, which includes selecting the appropriate level of autonomy, setting up the goal test(s), ii) accomplishing the goal, iii) checking the goal's accomplishment, iv) updating the plan with the new progress information. When the plan is complete, the loop ends.

4.8 Summary

This chapter presented the idea of Assigned Responsibility. This is a form of adjustable autonomy with pre-planned changes in levels of autonomy. Assigned Responsibility proposes to sacrifice some of the flexibility of more dynamic adjustable autonomy systems to improve automation reliability and human operator experience in known tasks where full automation is not currently achievable. Assigned Responsibility breaks up tasks into plans of goals and appropriately assigns responsibility for the accomplishment of these goals ahead of time to the human operator, the automation, or a mix of both. By doing this, it is intended that Assigned Responsibility will reduce workload and confusion in human operators, and provide clean boundaries for automation to operate in, easing the task of automating these individual goals, as well as supporting progressive automation of the whole task.

Section 4.4 presented an architecture for Assigned Responsibility, identifying the major components required in Assigned Responsibility systems, plan management and accomplishment monitoring, an architecture for which was presented in Section 4.5. Finally, Section 4.7 proposed a design for an Assigned Responsibility system.

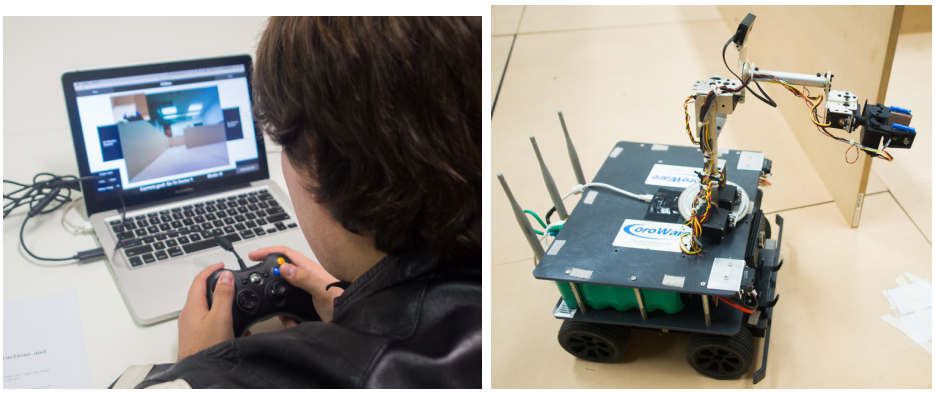
Chapter 5

Implementation

5.1 Overview

Before the idea of Assigned Responsibility can properly be evaluated, human operators have to be able to make use of it, meaning that a prototype Assigned Responsibility system has to be created. This chapter describes the implementation of the system designed in the previous chapter (Shown in Figure 4.9). As with all robotics projects, this implementation consists of hardware and software parts acting together. Seeing as Assigned Responsibility itself is a software problem, the hardware components are primarily based on off-the-shelf parts, modified to support the requirements of the software. The software that runs on this hardware platform was built from the ground up for this implementation.

Section 5.2 presents the hardware platform built to support the system, detailing the robot as well as the control station used by the operators. Section 5.3 describes in detail the software that was implemented for this project. Each component is explained and validated. The system is then shown to fulfil the requirements for an Assigned Responsibility system as described in Chapter 4.



(a) Laptop and Controller. **(b)** The Robot

Figure 5.1: Hardware used in the implementation. Communications between laptop and robot occur through the router mounted on the robot.

5.2 Hardware

5.2.1 Overview

The hardware in the project needs to provide support for all of the software used in the system. Due to the nature of remotely operated robots, this hardware has to be distributed in at least two subsections: the control station (here a laptop see Figure 5.1a), and the robot (Shown in Figure 5.1b). In this case, the software is divided into three subsystems, as shown in Figure 4.9: Operation, Management, and Execution, but only two distinct hardware components are needed to run the system, as both Operation and Management can run on the laptop, while the Execution subsystem is hosted directly on the robot. This mapping of software to hardware is not required for Assigned Responsibility, and simply one example of how such a system can be deployed.

5.2.2 The Laptop and Controller

The platform chosen to support the control end of the system consists of an off-the-shelf laptop and controller. The laptop (Pictured in Figure 5.1a) is a 13-inch Apple MacBook Pro, equipped with a 2.9GHz Intel Core i7 Processor, 8 GB of DDR3 RAM, and a 250 GB Samsung 840 Series solid

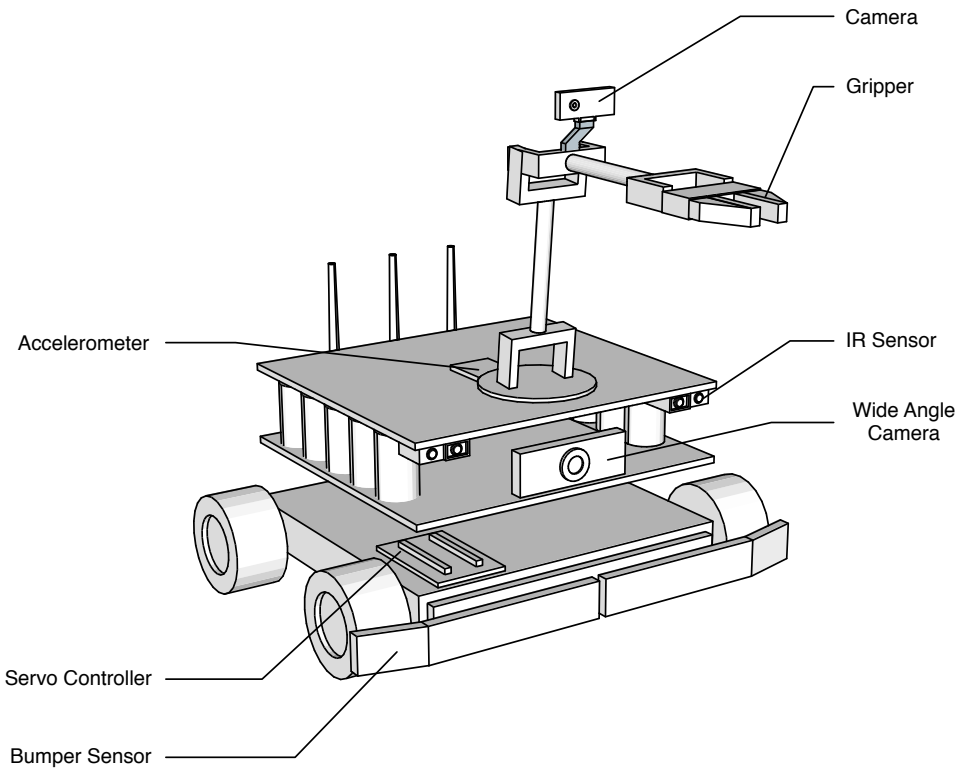


Figure 5.2: Major components visible from the front of the robot.

state drive. The laptop runs the Mac OS X operating system. Connected to the laptop via USB is an Xbox 360 controller used for most of the human input to the system, the remainder of which is done using the laptop's trackpad.

This setup was chosen for multiple reasons: i) it is a common setup for many teleoperation interfaces (see Section 1.3.3.c), ii) it facilitates development by providing a stable and powerful platform to run the system, and iii) it would be familiar to the operators that would use the system during its evaluation (All twenty of these operators did in fact report being familiar with game controllers (see Section 6.3.5)).

5.2.3 The Robot

The robot used in this system (Pictured in Figure 5.1b) is based on a Coroware Corobot, and has undergone significant modification. Figures 5.2, 5.3, and 5.4 depict the hardware mounted on the robot.

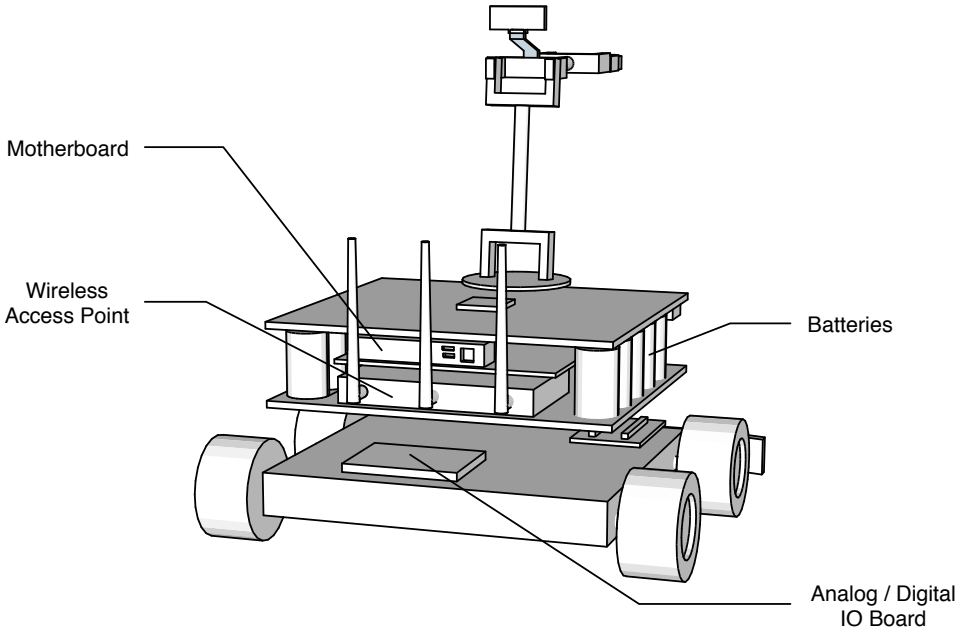


Figure 5.3: Major components visible from the back of the robot.

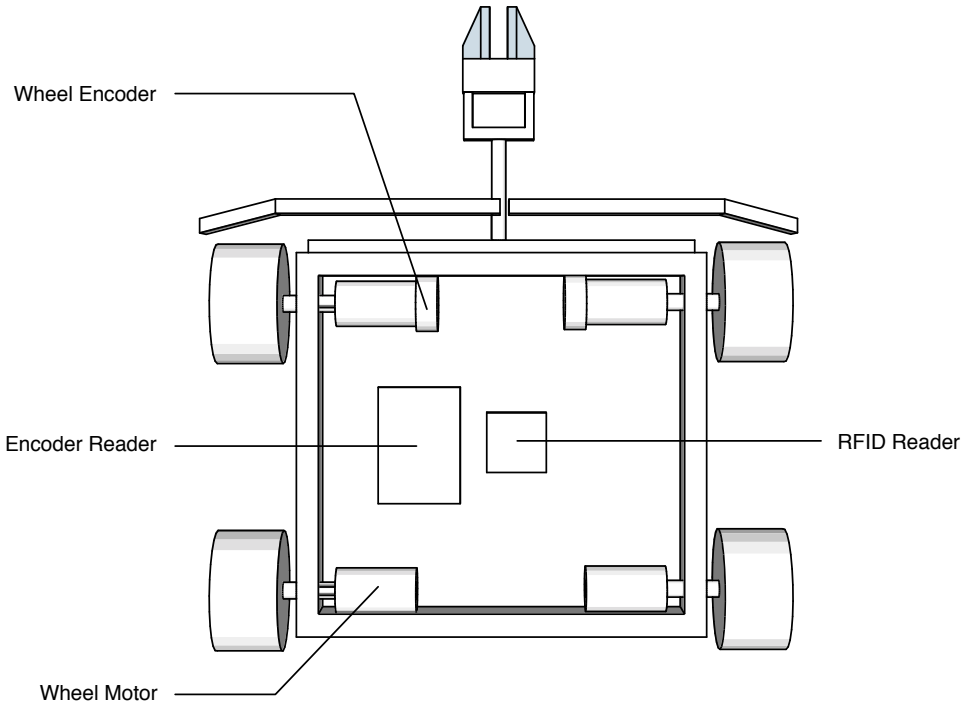


Figure 5.4: Major components visible from underneath the robot.

This robot design was chosen because it has the capabilities to fulfil navigation tasks, manipulation tasks, and observation tasks. This enables the robot to be used in usability tests of the implemented Assigned Responsibility on complex scenarios.

a. Processing

Mounted on the second highest layer of the robot is the Intel D2700MUD motherboard used to run the robot's operating system and software (see Figure 5.3). The robot runs on the Debian Linux operating system from a USB key. Most of the other components on the robot are connected to this motherboard through USB.

b. Networking

Mounted underneath the motherboard is a TP-LINK TL-WA901ND wireless access point which is broadcasting a WiFi network to allow devices to connect to the robot. This access point is connected to the motherboard via Ethernet. This router provides a 450Mbps transmission rate, sufficient to handle the incoming commands, outgoing sensor data, as well as the robot's video feeds. Commands to the robot were responded to with near zero measured latency, while the video feed was delayed by around 0.5 seconds. This video delay was not related to networking (as the response speed shows), but more likely caused by the robot's limited computational power, which resulted in a modest encoding speed for the two video feeds it had to process before sending on.

c. Sensing

The robot makes use of a variety of sensors to take in its environment. The list below details these sensors.

- Arm mounted camera (Figures 5.2 and 5.5). This small webcam is mounted in-line with the robot's gripper to assist the operator with lining up objects for pick-up or inspecting held objects. The camera is extracted from an Exoo Minocam webcam, and held in an aluminium case that was fabricated to allow it to be mounted.

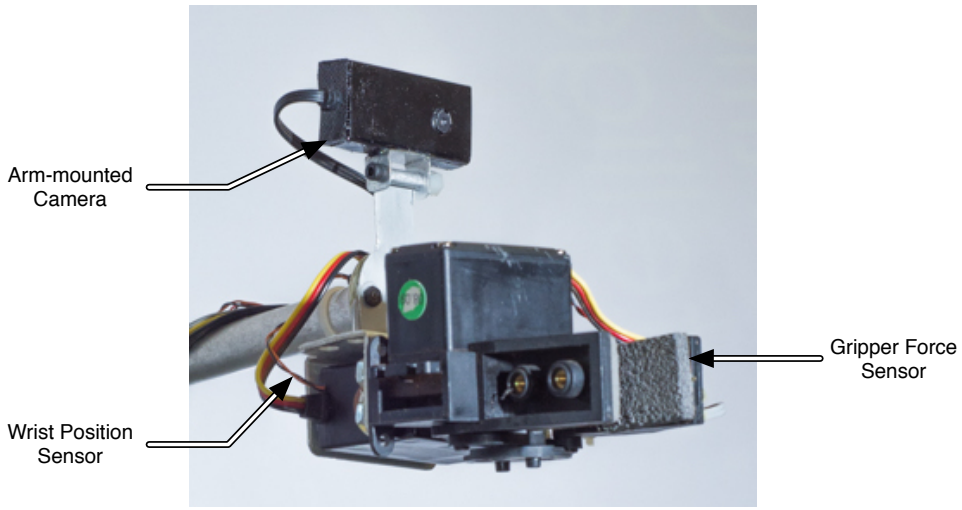


Figure 5.5: Sensors on the robot's end effector. Note that the force sensor is positioned between the grey cushioning pad and the plastic of the gripper. The position sensor works by running a current (through the wires pictured) to the servo's potentiometer and measuring the resistance.

- Front-facing camera (Figure 5.2). This camera is a Genius WideCam 1050 which was mounted to the front of the robot, providing a 120° view of the environment ahead of the robot.

- Infrared distance sensors (Figure 5.2). These SHARP IR proximity sensors are mounted to the front of the robot, one on each side, to provide a distance to obstacles, as well as allow an angle of incidence of the robot to a flat object to be calculated.

- Accelerometer (Figure 5.2). This Phidgets PhidgetSpatial 3/3/3 board is made up of a 3-axis accelerometer, gyroscope, and a compass, providing a variety of position change information. The implementation primarily makes use of the information from the accelerometer.

- Bumper sensors (Figure 5.2). These sensors are simple switches mounted into the base of the robot, triggering when an obstacle is hit.

- Wheel encoders (Figure 5.4). Rotational encoders on the two front

wheels are read using a Phidgets PhidgetEncoder HighSpeed 4-Input encoder reader. These sensors allow the distance travelled and manoeuvres done by the robot to be tracked.

- RFID reader 5.4. This sensor is a ID-12LA RFID module fitted to a Sparkfun USB RFID Reader. Mounted beneath the robot, it allows 125 kHz RFID tags on the floor to be read as the robot passes over them. These tags are placed at known locations and used to correct movement tracking errors from the wheel encoders.

- Arm position sensors (Figure 5.5). The servos used by the arm have been modified to allow their internal potentiometers to be read, allowing the angle at each joint to be determined.

- Gripper force sensor (Figure 5.5). The gripper is equipped with a force sensitive resistor used to measure the pressure applied to grasped objects.

d. Effecting

The robot has two means of making changes on its environment: the wheels, and the arm. All four wheels are powered by Hennkwell HG372 twelve volt DC motors. Steering is done through skid steering. The robot is equipped with a 5 degree-of-freedom arm, allowing it to manipulate objects in the environment. The arm is made of a rotational shoulder joint, a shoulder joint, an elbow joint, a wrist joint and a gripper, all powered by Hitec servomotors. Both the wheel and arm motors are commanded by software through the use of a Lynxmotion SSC-32 servo controller board, connected to the motherboard through a serial-USB converter. This servo controller board outputs pulse-width modulations to command the motors to a position (for the arm), or a speed (for the wheels).

e. Power

The robot can be operated while tethered to 12V DC power, but usually runs off of a pair of 6V 10 Ah NiMH batteries wired in series to provide the

12V required by the robot. The batteries can keep the robot operational for approximately two hours of use.

5.3 Software

5.3.1 Overview

While the hardware presented above is a fundamental part of this system, the core of the work presented here lies in the software that runs on this hardware. This software is divided into three modules as described in the previous chapter and pictured in Figure 4.9: operation, management, and execution. At any moment, the software must handle input from one or more operators, updates to the plan structure, monitor world states, and execute commands on the robot. This means these modules must function in parallel, somewhat independently of each other, as well as handling the fact that they are distributed in separate physical locations. Both this parallelisation and separation means the system needs a reliable method for communication between modules.

This section presents the software side of this implementation, describing each of the three modules as well as the communications network that links them. The components that make up the software side of the system are all written in the Python programming language (Version 2.7) (Foundation, n.d.).

5.3.2 Communications

The communications between the components of the system are mapped out in Figure 5.6. These take two forms, remote object calls, and video streams. To handle object calls between the modules, the Pyro (Python Remote Objects) Python library is used (de Jong, 2016). Pyro allows Python objects to make method calls available across processes and across machines through the use of name servers. This way, modules can communicate with each other directly by calling each other's methods. This system makes use of two name servers, one on the robot, and one on the laptop as shown in Figure 5.6.

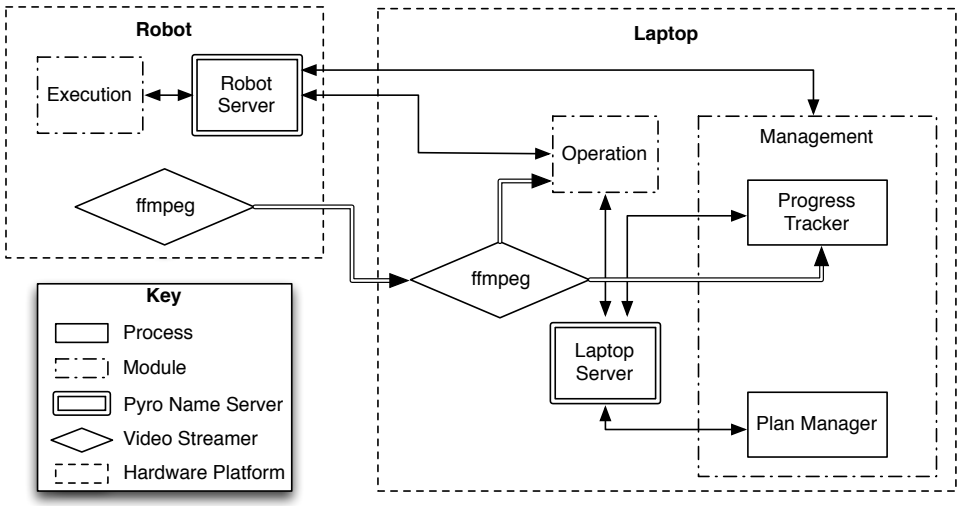


Figure 5.6: Communications between elements of the system. Single stroke arrows designate remote object calls, double stroke arrows represent video streams.

The robot name server publishes the robot client's methods, primarily to allow the operation module to send commands to the robot, and for both the management and operation modules to retrieve sensor information from the robot. For example, the operation module calls the *send_claw_pos* method on the robot to set the gripper's position. This function takes the arm position as an argument. Methods called through Pyro can also return data, allowing information like sensor readings to be retrieved. The laptop-based name server is used to publish methods from the plan manager component of the management module. Its primary purpose is to allow the other modules to read from and modify the plan structure.

The only communications occurring outside of this setup are the video streams from the two cameras mounted on the robot. These two streams are captured and sent over the wireless network directly to the laptop through the use of the ffmpeg software. These streams are then accessed as needed by the laptop-based modules. To prevent clutter in diagrams, the remainder of this chapter will describe communications between modules as if they occurred directly. The use of name servers is implied.

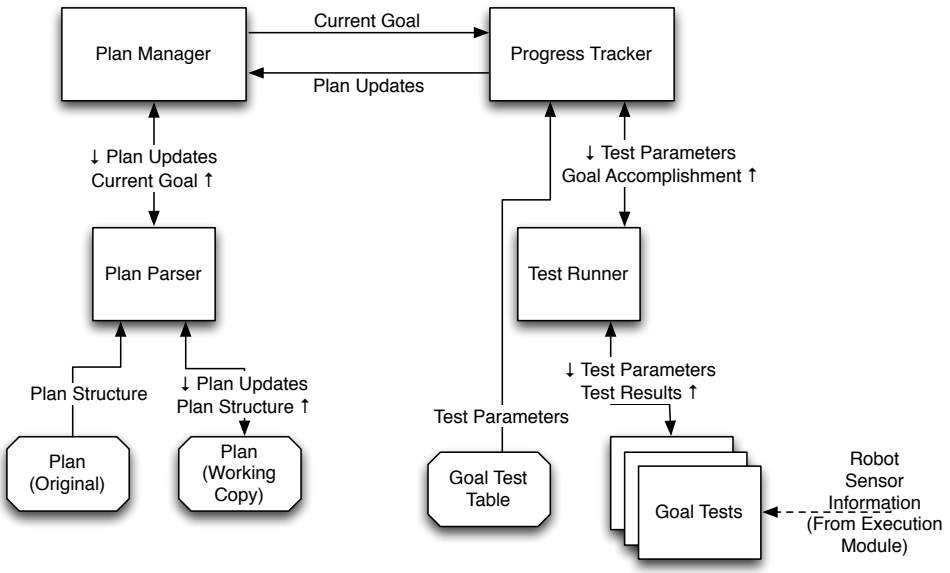


Figure 5.7: Data flows in the management module.

5.3.3 Management

The management module is responsible for two major tasks: keeping track of progress through a task, and maintaining the plan structure used to store that progress up to date. To this end, the management module is split into two processes that operate in parallel: the plan manager and the progress tracker. Figure 5.7 illustrates the structure of the management module, which is explained below.

a. The Plan Manager and the Plan

The plan manager serves as a front-end to the plan data structure, calling on the plan parser to read from and make modifications to the plan. The plan manager is launched with the name of the plan as an argument. When launched, the plan manager copies the original plan file to create a working plan file. This preserves the original, while allowing the progress of the current undertaking to be stored. In case of an interruption, the working plan can be resumed, or the original plan can be re-started.

The plan is stored in a specialised XML structure, the schema definition for which is visible in Appendix A.1. Figure 5.8 shows an example plan

```
<plan_tree>
        <goal goal_type="Parent">
                <name>Close All Valves</name>
                <status>Started</status>
                <subgoal goal_type="Parent">
                        <name>Close Valve 1</name>
                        <status>Started</status>
                        <subgoal goal_type="Achieve">
                                <name>Go To Sector 1</name>
                                <status>Accomplished</status>
                                <assignment>A/H</assignment>
                        </subgoal>
                        <subgoal goal_type="Achieve">
                                <name>Search For Valve 1</name>
                                <status>Started</status>
                                <assignment>H/A</assignment>
                        </subgoal>
                        <subgoal goal_type="Achieve">
                                <name>Close Valve 1</name>
                                <status>Not Started</status>
                                <assignment>A</assignment>
                        </subgoal>
                </subgoal>
                [...]
        </goal>
</plan_tree>
```

Figure 5.8: Sample XML plan structure.

written following this schema. This file specifies the structure of the plan, providing the relationships between goals through the use of nesting, as well as additional information about these goals, such as their current state (One of: *Not Started*, *Started*, or *Accomplished*), and the level of autonomy (Referred to as the *control mode* in the system) they are to be accomplished in.

XML was chosen as the language to store the plans in because of its syntax, which suits tree-style structures due to the nested nature of XML. XML is also easily parseable even when using custom schemas, an important aspect that saved development time. Finally, schemas themselves prove useful to ensure plans written for the system have a common set of rules, ensuring that anyone can write plans that will work in this system as long as they follow these rules.

b. The Progress Tracker

The progress tracker is charged with tracking the progress of the robot through the plan. This tracking is performed on a goal-by-goal basis, which requires a wide variety of goal tests to be performed. To support this variety, the progress tracker is simply responsible for providing the test runner with the information needed to run the appropriate tests for the current goal, as well as requesting the plan's update when a goal is found to be accomplished.

To this end, the progress tracker first retrieves the name of the current goal from the plan manager. Using that name, the progress tracker consults the goal test table, a CSV file that stores goal names and associated goal tests, as well as the desired goal state information used to test the goal's accomplishment (An example is shown in Figure 5.9). This file specifies which tests have to be successful before a goal can be considered to be accomplished. There can be one or more tests for each goal. Each test is given a confidence value between 0 and 1 to specify how trusted a goal test is. This value is used when determining the overall outcome of the accomplishment checking: the confidence values for successful tests are added, and if the total value equals or exceeds 1, then the goal is considered to be accomplished.

```
# Format: "Goal Name", "Test Type 1", "Desired State 1", Confidence between 0—1, "Test Type
     n", "Desired State n", Confidence between 0—1,
"Go To WP1", "surf", "images/WP1.jpg", "0.7", "gps", "—31.754813, 115.979397", "0.3"
"Report WP1 State", "log_parser", "logs/log.txt, Received", "1"
"Go To WP2", "surf", "images/WP2.jpg", "0.7", "gps", "—31.754813, 115.979397", "0.3"
"Report WP2 State", "log_parser", "logs/log.txt, Received", "1"
"Go To WP3", "surf", "images/WP3.jpg", "0.7", "gps", "—31.754813, 115.979397", "0.3"
"Report WP3 State", "log_parser", "logs/log.txt, Received", "1"
"Go To WP4", "surf", "images/WP4.jpg", "0.7", "gps", "—31.754813, 115.979397", "0.3"
"Report WP4 State", "log_parser", "logs/log.txt, Received", "1"
"Go To WP5", "surf", "images/WP5.jpg", "0.7", "gps", "—31.754813, 115.979397", "0.3"
"Report WP5 State", "log_parser", "logs/log.txt, Received", "1"
"Go To WP6", "surf", "images/WP6.jpg", "0.7", "gps", "—31.754813, 115.979397", "0.3"
"Report WP6 State", "log_parser", "logs/log.txt, Received", "1"
"Return to Base", "gps", "—31.754813, 115.979397", "1"
```

Figure 5.9: Example goal test table CSV file.

```
from re import split
from Pyro4 import locateNS, Proxy

class NavCheck(object):
    def run(self, boundary):
        success = False
        boundaries = split(", ", boundary)
        for b in boundaries:
            b = int(b)
        # Open comms with the pyro server
        name_server = locateNS('localhost', 9090)
        uri = name_server.lookup("manager")
        manager_handler = Proxy(uri)
        pos = manager_handler.get_message('POS')
        if pos is not None:
            # check to see if we are inside the boundary
            if int(boundaries[0]) < pos[0] + 225 < int(boundaries[2]) and int(boundaries[1])\
                    < pos[1] + 225 < int(boundaries[3]):
                success = True
        return success, ""
```

Figure 5.10: Goal test for navigation goal.

This confidence system allows for an approximation of boolean logic with various configurations of test successes able to result in an overall goal accomplishment. For example, a goal with only one test needs that test to have a confidence value of 1. A goal that requires two tests to succeed (boolean AND) must give those tests a cumulative value of 1: such as 0.5 and 0.5. A goal that requires one test to be successful as well as either of two other tests (x AND y OR z) could use these values: 0.6, 0.4, 0.4 etc.

The progress tracker provides a list of the tests to run to the test runner, as well as what qualifies as a test success. Once the test runner returns the cumulative confidence value of the successful tests, the progress tracker determines if the goal is accomplished. If it is, a plan update is requested. If not, the tests are re-run.

The tests themselves are python modules that have to respect a particular input and output format, but otherwise vary based on the type of information they have to process. Figure 5.10 showcases a simple navigation based goal test, which checks if the robot falls within the boundaries of a desired location. The boundaries are provided by the goal test table, while the robot's position is gathered directly by the test. Each test is expected to return the success or failure of the test through the use of a boolean (True = success, False = failure), as well as an optional logged message that can be used for debugging or simply returning more information about the test results.

The operation module's primary goal is the issuing of commands for the robot. To issue these commands, the module leverages a mix of human input and automation. As a result, the module is structured around two main processes running in parallel (shown in Figure 5.11): the human interface (HI), and the automation manager.

5.3.4 Operation

The human interface is primarily charged with providing the human operator with feedback from the robot, as well as allowing the operator to command the robot. This requires the human interface process to interact with other processes. The controller reader captures input through the

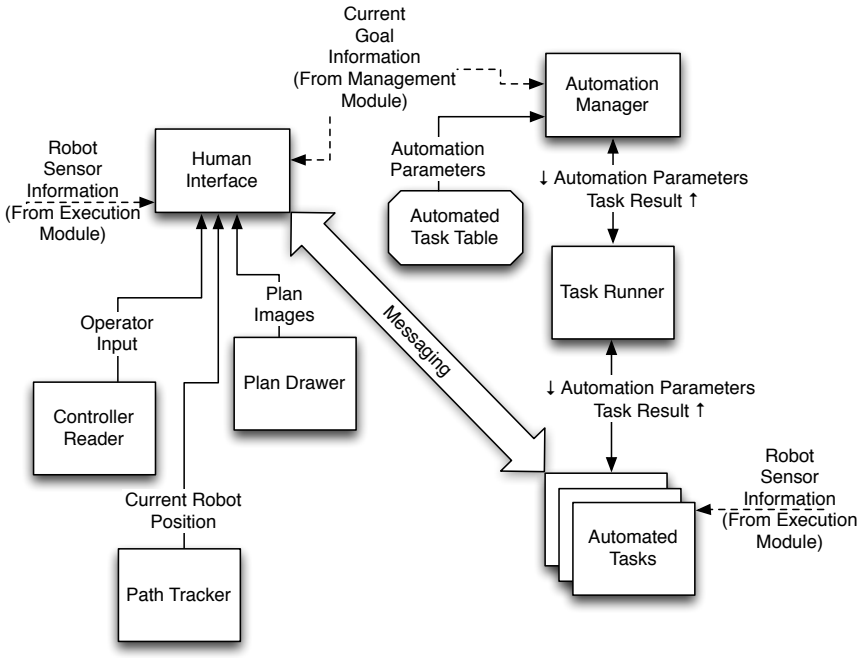


Figure 5.11: Data flows in the operation module. The operation module is responsible for managing human and automated commands, merging them if needed.

controller (mouse input is captured directly by the HI). The path tracker polls the robot sensors to keep track of the path travelled by the robot, information which is then used by the HI to draw a map of that path. Finally, the plan drawer generates the visual representations of the plan used by the interface.

The automation manager ensures that if automation is required, the correct automated task is running. To do this, the manager compares the current goal to its table of automated tasks, passing the relevant parameters to a task runner if there is a match and automation is required. The task runner then selects the correct automated task from its library of automated tasks, and passes the relevant parameters to it at launch. Messaging between the HI and the automated task exists for mixed control situations, when human and automation have to work hand in hand.

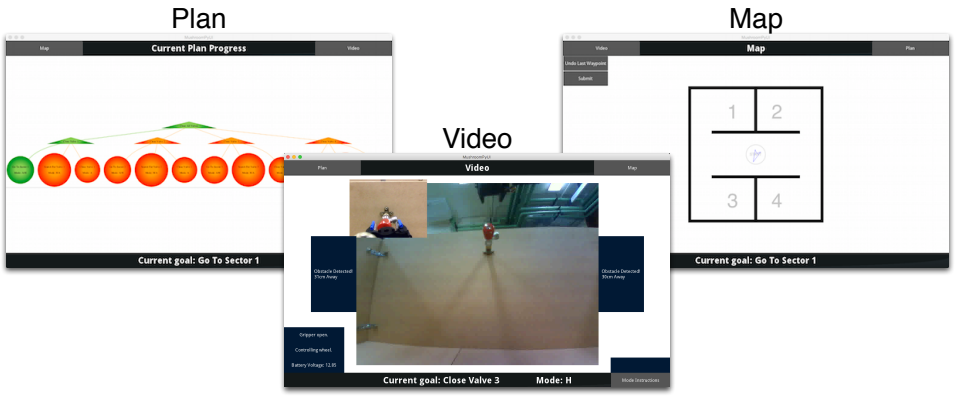


Figure 5.12: Screens available to the operator.

a. Human Interface

The human interface provides the human operator with the ability to issue commands to the robot, either directly, or through high-level commands to the control automation. Direct control is handled for the most part through the use of a game controller, while high-level commands are communicated through direct interaction with the system's interface using a mouse. The control mode (level of autonomy) of the system changes what inputs are available to the human operator, with the controller typically only used in low automation modes for example.

The other role of the interface is to display system information to the operator. This information ranges from sensor readings from the robot, to progress through the plan. The interface is divided into three screens, as shown in Figure 5.12. The operator can freely shift between these screens by using the buttons in each top corner. At the bottom of each screen, the current goal is displayed.

The first screen, *Video*, is the default screen. It displays tiled streams from both cameras mounted on the robot, as well as distance sensor information (on either side of the video stream, for the left and right side readings respectively), battery levels, and whether the operator is commanding the robot's wheels or arm.

The *Map* screen is used to show the path the robot has travelled. This information is compiled by the interface from the robot's sensor data. To

add context to the path, it is possible to provide the interface with a map of the location the robot is operating in. The path is then drawn over this map. The screen also presents the same sensor information as the video screen, minus the video, to allow the robot to be driven from this view. The map screen is also used to communicate high level navigation commands to the automation in the form of waypoints.

The *Plan* screen is used to show the current state of the plan: what the current goal is, what has been accomplished, as well as what control mode goals are to be performed in. This screen provides context to the goal being currently accomplished, allowing a human operator to return after a break and understand what has happened.

b. Automation Manager

The second half of the operation module, the automation manager, functions similarly to the progress tracker described above. The manager is only responsible for selecting the appropriate automated module based on information provided in the plan: the current goal, and the current control mode. This information is passed on to the task runner which actually launches the task itself. In a similar fashion to the goal tests used by the progress tracker, these tasks fit an input/output template to allow them to be called by the same process, but are free to act as needed otherwise.

For the most part these tasks simply request appropriate sensor information from the robot, but in the case of human/automation collaborative control, the automated task communicates with the human operator.

5.3.5 Execution

The execution module is responsible for two roles in the system: execute the commands received from the operation module, and provide access to sensor information to the rest of the system. The module (Shown in Figure 5.13) is composed of the robot server, responsible for handling requests made by the other modules, and a multitude of handlers, pieces of software responsible for communicating with the robot's hardware. One of these handlers is

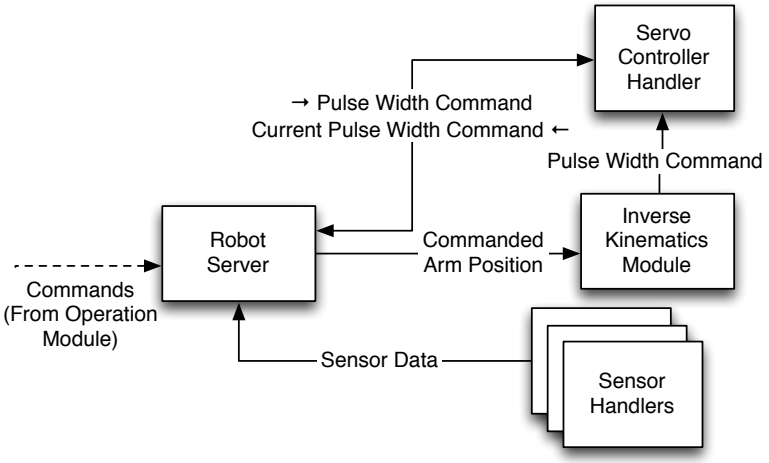


Figure 5.13: Data flows in the execution module.

responsible for communications with the servo controller, hardware used to command the servo motors of the robot's arm and wheels. The other handlers are used to communicated with the sensors mounted on the robot. The robot server serves as a front-end for these handlers, allowing external processes to query sensors and control the robot's actuators.

The only real processing the robot server does is related to the control of the robot's arm. Rather than specifying joint positions directly, arm commands issued from the operation module come in the form of 3D coordinates that represent a desired position for the gripper. This point is calculated using the joystick movements the operator makes; from a set start position, each time the operator moves the joystick, a translation corresponding to that movement is done on the point. If the joystick points down, the point moves down etc. The robot server then calculates the arm joint positions (Using inverse kinematics) required to get the gripper to these coordinates. These joint positions are then sent to the servo controller handler for execution.

5.4 Sensor Evaluation

5.4.1 Overview

The system uses sensors mounted on the robot to both allow control of the robot, as well as track goal accomplishment. Before sensors can be used for either, their capabilities have to be measured, and their suitability for goal testing established. While some sensors are mounted on the robot at all times, some sensors described below (Such as the GPS sensor) are only mounted on the robot when required.

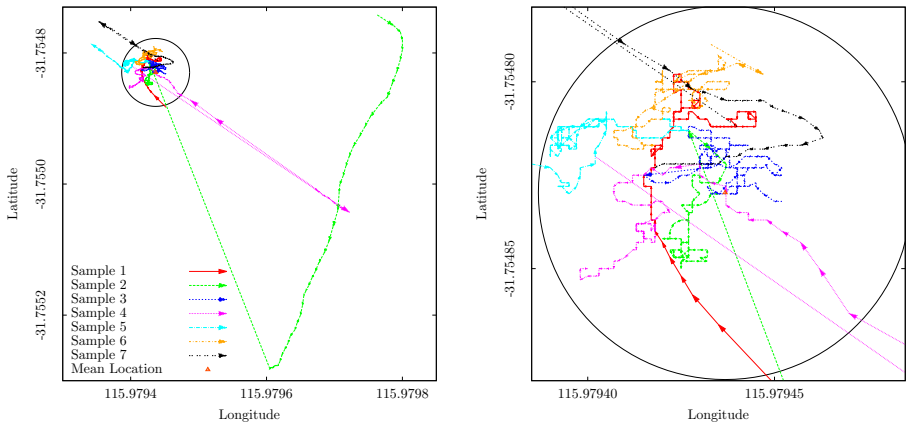
This section presents information regarding the robot's sensors, detailing the data they return, their limitations, as well as how they could be used to test the accomplishment of goals.

5.4.2 Location

Just like other goals set to the robot, the accomplishment of location goals must be checked. The system makes use of two location tracking methods for this purpose. While location tracking is a well described problem, it is important to evaluate the accuracy of the available methods before relying on them, as some methods may be not be well-matched to their applications, for example: some only work outdoors, while others are only accurate enough to track large objects.

a. GPS

The robot's latitude, longitude, altitude, and velocity can be obtained from a PhidgetGPS board with a manufacturer reported best-case circular error probability of 2.5m. To gauge its accuracy, the sensor was tested in optimal conditions, an open park under clear skies. Seven series of data points, collected for roughly one minute each, were recorded over a period of three days from the same location. These sets are plotted in Figure 5.14. This totalled 1059 individual readings, which were found to diverge on average from a calculated mean position by 3.53 meters. As Figure 5.14a shows however, some readings diverge by up to 40 meters. Most of these large errors occur early in the sample, as the calculated location tends to converge



(a) Overall view of the points collected.

(b) Zoomed-in view of the area directly surrounding the mean coordinate.

Figure 5.14: Baseline readings using the GPS sensor. Readings taken in seven groups over three days in ideal conditions with the sensor in a fixed position. A four meter radius circle was centred on the mean coordinate

towards the mean position after a few seconds, but some occur later on in the sample as, presumably, the sensor loses contact with one or more satellites. These eccentric readings tend to be in the minority, and can be mitigated by calculating a mean position using several seconds of readings.

These results show that this particular GPS sensor is too inaccurate for precise ($<$ 2m) position calculations, but is suited for tests requiring less precise positional information, such as arriving in an area rather than at a specific point. These findings suggest that goal tests that make use of information from this GPS sensor need to be fairly lenient, and allow a good margin of error for the positional information provided by the sensor. A threshold of four meters (The 3.53 mean distance found above rounded up) should be sufficient as the criteria for success in the GPS test. This four meter radius circle is plotted in Figure 5.14. Figure 5.14b shows the readings reported in the test that would be found inside that circle.

b. Odometry

An alternative to absolute position measurement by satellites is to use sensors mounted on the robot to calculate relative position changes. The

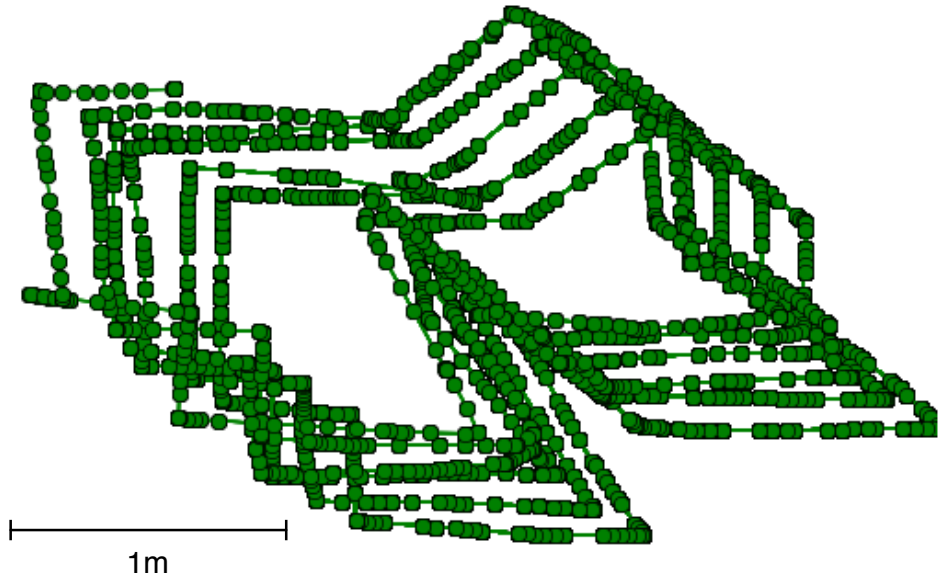


Figure 5.15: Robot path drawn using odometry. Note the errors accumulating over time.

information used for this process comes from the wheel encoders and the accelerometer. For each measurement, the wheel rotations provide translation distance, and the accelerometer provides the angle of that translation. By accumulating these measurements, it is possible to record the path travelled by the robot. This method is imprecise however, as errors can easily accumulate over time. Figure 5.15 maps the path undertaken by the robot during a test. This map shows the systematic errors as the recorded path the robot travels slowly drifts over time even though the robot is traversing the same circuit.

These accumulated positional errors are the result of a multitude of factors, including sensor inaccuracy, and oversimplification in the modelling of the robot's movement. For example, when skid steering, the robot might slip and not exactly turn on the spot, whereas for simplicity's sake, the algorithm assumes a perfect turning manoeuvre.

To stop the errors from accumulating, it is necessary to occasionally correct the position of the robot using a landmark with a known position. The odometry can then be used to track movement between these resets,

providing sufficient accuracy for most tasks. This also offers another benefit: if the human operator or the automated controller realise that the robot is not where the odometry claims it is, they can search for one of these landmarks to reset their position before continuing with the task.

5.4.3 Vision

The robot is equipped with two webcams: one body-mounted wide-angle camera and an arm-mounted camera. While the images returned by these are useful as-is for the human operators, they need to be processed to extract information meaningful to automated systems. This section details three strategies used to process and analyse the webcam images for use by the automated controllers and the goal tests. These techniques are off-the-shelf algorithms that were implemented for this system, and as such the interest in this section lies in how these techniques can optimised and how much they can be relied on for goal accomplishment detection.

a. Feature Detection

This first strategy tackles the need to recognise complex features in the robot's environment. These could be signs, panels, or any other objects of interest that could not be detected using other techniques. This recognition of landmarks and task states is achieved by the Speeded-Up Robust Features (SURF) (Bay, Ess, Tuytelaars, & Van Gool, 2008) module from the Python OpenCV library. This is a scale and rotation invariant image matching algorithm, which can be used to compare the camera input to a reference image based on correspondences of interest points in the two sources.

The behaviour of SURF was studied in our laboratory under systematic adjustments to two key parameters, *nOctaves* and *hessianThreshold*, in order to optimise its performance. *nOctaves* sets the number of Gaussian pyramid octaves used for matching, which affects the grain size of detected features. *hessianThreshold* sets the acceptable feature strength for retention of matched interest points.

The algorithm was modified to return a confidence value expressing the number of matched interest points in real time. Table 5.1 shows number of

Hessian T./ nOctaves	300		400		500	
	Target	No Target	Target	No Target	Target	No Target
2	21.89	3.943	20.78	4.525	19.71	5.15
3	19.24	5.257	23.16	4.33	19.78	4.46
4	21.91	3.567	24.33	5.8	19.33	3.47
5	24.19	3.767	25.46	4.237	17.63	3.34
6	23.26	3.607	23.89	4.81	20.73	5.11

(a) Key points detected for each nOctave / hessian threshold combination.

Hessian T./ nOctaves	300	400	500
2	5.55	4.59	3.83
3	3.66	5.35	4.43
4	6.14	4.19	5.57
5	6.42	6.01	5.28
6	**6.45**	4.97	4.06

(b) Signal-to-noise ratio for each nOctave / hessian threshold combination

Table 5.1: Results of the SURF algorithm tests. These results serve to calibrate the test for a particular camera and target object. As such this table is more useful as an example of the calibration process than as an idea setting. The goal is to find the combination of nOctaves and Hessian threshold that returns the largest difference between the number of key points matched when looking at the target and the number of key points matched when looking away from the target. If this difference is high then the detection rate of false positives is likely to be lower.

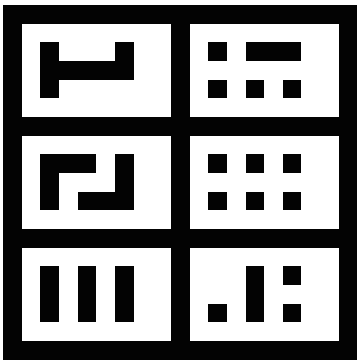
observed matched points and signal-to-noise ratios for 2-6 octaves and Hessian thresholds over the recommended range of 300-500. We chose *nOctaves of 6* and a *hessianThreshold* of 300 because these returned the best signal-to-noise ratio and a good number of points on our target object.

A goal test using this algorithm would simply observe the returned confidence level. If above a certain threshold, the test would return positive. Setting this threshold to the right level is an important task, and should not be done generally, but rather on a case by case basis, because some landmarks simply possess more features detectable by the algorithm than others. For feature-rich landmarks, the threshold can be set high with confidence, while less feature rich landmarks will require lower thresholds to be detected. These lower thresholds can be an issue, as the lower the threshold, the more likely that background noise will trigger the test. In this case, it may be necessary to supplement the goal test with data from another sensor.

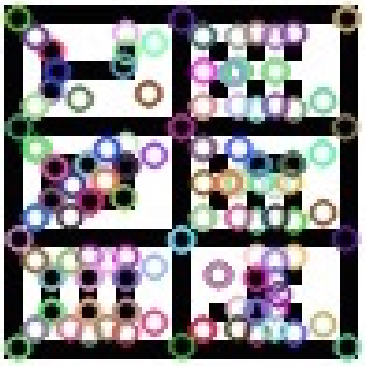
b. Fiducial Tracking

Fiducials are feature-rich markers that can be placed in an environment to help with visual processing of a scene (Lepetit & Fua, 2005). Fiducials are usually placed on objects of importance to improve tracking and orientation estimation of those objects. While some fiducial tracking systems make use of complex algorithms that help differentiate similar looking fiducials (Kaltenbrunner & Bencina, 2007), the approach used here is simpler and brute-force based.

Using the Python OpenCV library, the fiducial tracking algorithm written for this implementation performs a series of steps on each frame provided by the webcam (see Figure 5.16). The fiducial tracker begins by loading the reference image of the fiducial (Figure 5.16a) and detecting keypoints in the image using the SIFT library (Figure 5.16b). The tracker then opens the webcam feed and starts analysing each frame. The analysis occurs in three steps: 1) the frame is read (Figure 5.16c), 2) the frame is thresholded using adaptive thresholding (Figure 5.16d), and 3) keypoints are detected. The tracker then matches the keypoints found in both the

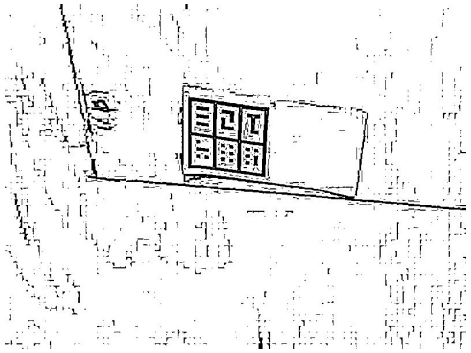


(a) The fiducial.

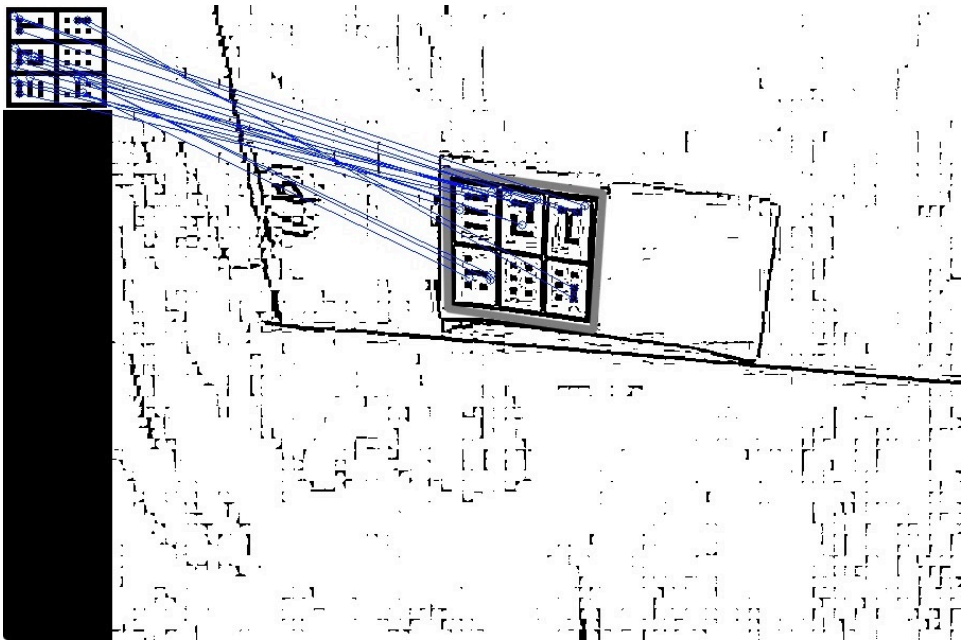


(b) Detected keypoints.

(c) The frame from the webcam.



(d) After adaptive thresholding.



(e) Locating the fiducial in the image by matching the key points detected in the image to the key points detected in the reference fiducial.

Figure 5.16: Steps in the tracking of a fiducial marker in a webcam feed.

fiducial and the frame to determine both if the fiducial is present (using a threshold), and the orientation of the fiducial using homography (Figure 5.16e).

c. Colour Tracking

Colour tracking is the simplest of the three vision-based detection approaches presented here, but can be very effective in the right environment. It consists of isolating a colour range in the robot's camera images, and calculating the centre of gravity of the resulting shape. This is done using the Python OpenCV library, using simple built-in tools like image thresholding (to isolate the colour patches) rather than complex algorithms like SURF and SIFT as used above.

As a result, this technique consumes much less processing power than the previous two techniques, but is only useful if the target object is distinctively coloured, and in an environment lacking that colour. It is possible to limit the range of acceptable colours to mitigate some of this drawback however. Goal tests using this method will be based on a threshold dictating the minimum acceptable size of the resulting patch of colour.

5.4.4 Force Sensing on the Gripper

To achieve its goals, the robot must manipulate the environment, and sensors mounted on the robot's arm can provide some indication of goal success. Part of this is done through the use of a force sensitive resistor mounted in the robot's gripper as described above. This sensor is used to measure the pressure placed on the inside of the gripper, indicating if the gripper is holding on to an object, or if it is open or closed. The analog signal is linear with the applied force and is also? used to measure how hard the gripper is pressing on an object.

Another kind of force sensing is also available on the gripper through the use of a potentiometer on the gripper's wrist motor. This allows the system to determine whether the wrist motor is encountering resistance when turning. At launch, the system calibrates the wrist by rotating it while empty, to get a baseline function: commanded position (current pulse

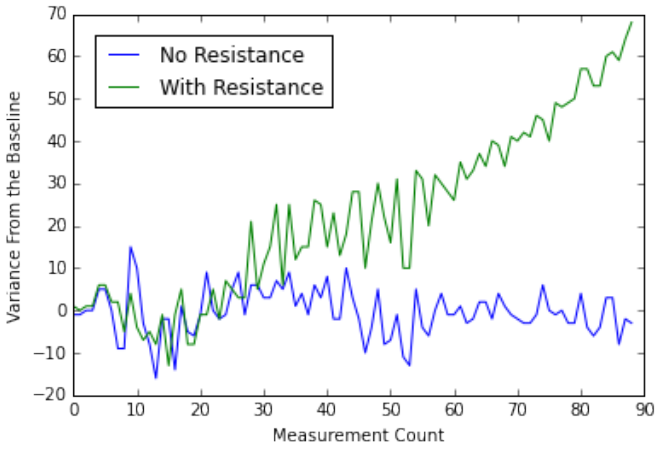


Figure 5.17: Force sensing using commanded position vs actual position. A set of baseline measurements were recorded, then the gripper was rotated twice, once with no resistance imposed on that rotation, and once with resistance applied. Variance from the baseline indicates whether the wrist rotation is diverging from the baseline measurement or not. Measurements are made for each increment (roughly a degree each) the wrist is moved by.

width modulation sent to the servo) vs actual position reading (converted to a pulse width equivalent from the servo potentiometer). This is done by commanding the wrist motor to rotate increasingly to the right by about a degree each time, and compare these increasing commands against the readings from the potentiometer. With an empty grip, these values should be close to identical, so a commanded move of a degree should result in an actual move of a degree. When rotation becomes harder, the difference between commanded position and actual position rises, allowing the detection of the force being applied. Figure 5.17 shows this effect; the blue line represents an empty rotation, which stays close to the baseline (does not deviate far from zero) throughout the range. The green line displays a gradual deviation from the baseline, a clear sign something is hampering the rotation.

In the experiments described later (see Chapter 6), the robot is tasked with closing a set of taps. Using a combination of both of these force sensing strategies can be used to detect when these taps are closed. The gripper sensor can be used to determine if the gripper is closed on the tap, and the wrist sensor if the gripper's rotation is unhindered (the tap is not closed

yet), or encountering resistance (the tap is closing or closed).

5.5 Functional Testing

5.5.1 Overview

This section presents the testing undertaken to ensure the implementation presented above is functional and ready for the experiments described in Chapter 6. This section is divided into two functional tests, the first covering testing of the management module, the second a series of small trials testing the rest of the system.

5.5.2 Functional Test 1: Robot platform testing in the Australian desert.

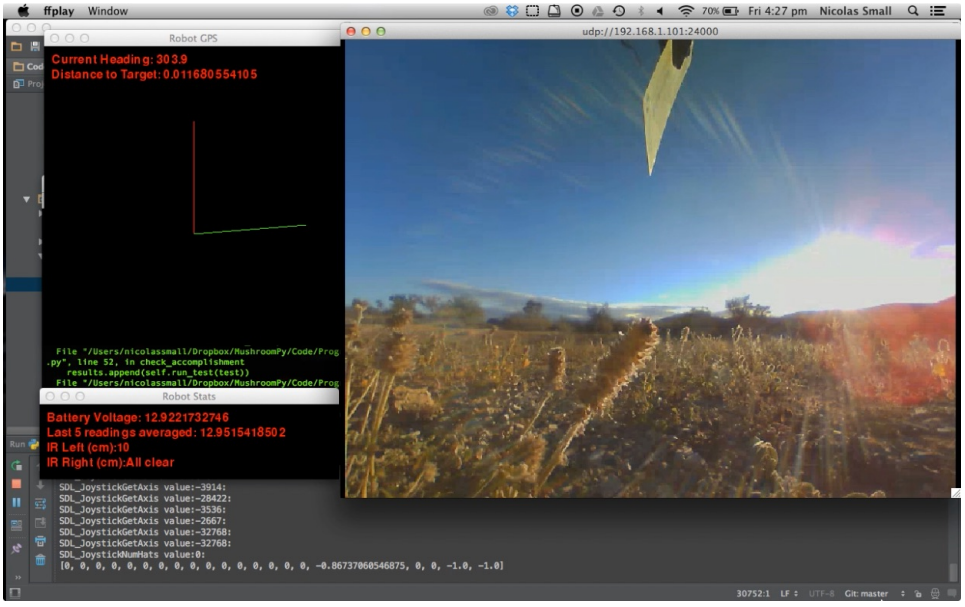
The robot was taken to the Arkaroola desert station in central Australia to take part in the Arkaroola Mars Robot Challenge (Mann et al., 2015). The robots tackled a series of tasks composed of six standard robot tests developed by the National Institute of Standards and Testing (NIST) as well as two operational tests simulating Mars surface missions. This robot was not designed to be Mars-capable, but the challenge provided a useful opportunity to test it for ruggedness in a demanding environment. Figure 5.18 provides images from the challenge. The challenges highlighted issues on the control end and the robot end of the system, prompting several changes in this implementation.

As highlighted in Figure 5.18c, the robot had issues with tipping over when traversing some of the rough terrain, and required the operator to carefully pick a path using the video feed to avoid tipping the robot over. The only camera mounted on the robot at the time, the wide angle (120°) body-mounted webcam (the view from which is shown in Figure 5.18d) proved useful to avoid rocks and other raised obstacles, but was of little use for spotting cracks in the ground. This, coupled with the difficulties faced by the operator in performing search tasks, led to the addition of the arm-mounted camera described in Section 5.2.3. This narrow field-of-view (30°) camera provides an adjustable viewpoint (through moving the arm) as well

(a) The Arkaroola quarry. This site offered varied difficult terrain useful for stress testing robots.

(b) The robot on the pitch/roll ramps during the NIST benchmarking.

(c) The testing put the robot through its paces.



(d) This early implementation of the teleoperation interface is far more basic than the final result shown in Figure 5.11. Lessons learned during these experiments helped shape that final product. The object at the top of the image is a geologist's photographic scale held in the gripper.

Figure 5.18: Photographs of the environment and tests, and interface used in the Arkaroola robot experiments.

Figure 5.19: Photograph of the maintenance imaging scenario test site. Waypoints are highlighted.

as a more zoomed-in view that eased searching tasks.

5.5.3 Functional Test 2: Progress Tracking and Plan Management

This test focuses on the management component of the system, to test if the progress tracking can consistently track the execution of the plan, and if the plan management component updates the plan graph with this progress information. While these are the primary objectives, this experiment also incidentally tests the functionality of the human interface, as the experiment calls for the robot to be operated by a human operator. The experimental scenario and tasks required of the robot during this experiment are detailed below.

a. The Maintenance Imaging Scenario

Automatic regular maintenance of physical equipment requires a mobile robot to periodically visit a number of key worksites, where the robot may perform tasks such as photography, gathering sensor data on environmental conditions, physically probe the integrity of surfaces, joints or attachments, remove panels and/or to change out faulty components (Mann, 2008).

In this experimental scenario, a robot is teleoperated around a series of four worksites, aligning itself close to each one in turn so as to be able to photograph important objects. Figure 5.19 shows the test environment.

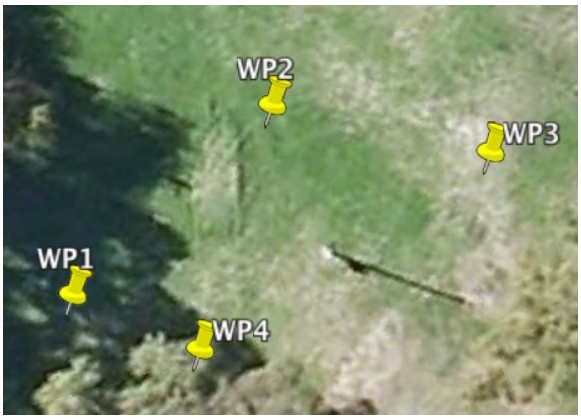


Figure 5.20: Reference GPS coordinates for the maintenance imaging scenario plotted in Google Earth.

Figure 5.20) shows the waypoints as plotted in Google Earth, offering an overhead view of the test site's arrangement.

The goals of being at each photograph point are defined by a bounding circle of GPS coordinates associated with the worksite and an image for matching using the vision system. Figure 5.21 shows the task as described by these goal tests, listing the coordinates and the visual target for each waypoint. A goal is met only if the robot's current GPS coordinates fall within the specified range and the vision system is suitably confident of a matching visual image.

b. Experimental Setup

This experiment makes use of a stripped-down teleoperation interface (used solely to control the robot's movements, with no automation present). The plan used in the experiment is shown in Figure 5.22. Note the absence of control modes in the plan; this experiment only used the teleoperation (T) mode, and did not test changes in levels of autonomy.

All of the goals in the plan require the robot to go to a particular waypoint, which required the ability to locate the robot so as to confirm the accomplishment of those goals. This Goal Accomplishment Tracking used a GPS sensor mounted on the robot (detailed in Section 5.4.2), as well as analysis of the video feed coming from the robot using the SURF

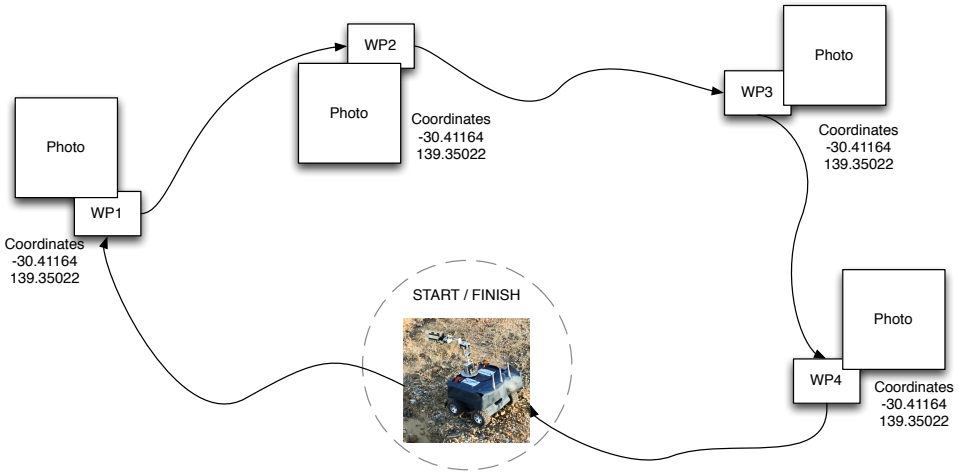


Figure 5.21: Diagram of the Maintenance Imaging Task. For each goal to be satisfied, both sensor tests must return positive: the robot's current coordinates must fall within a bounding circle of reference GPS coordinates, and the reference image must be matched within the robot's video stream.

feature detection algorithm described in Section 5.4.3. Each goal is considered accomplished when both sensors report a match on the desired world state (these requirements are recorded in Figure 5.23). Although the GPS sensor seems sufficient to confirm the location of the robot, the data presented in Section 5.4.2 shows that the GPS unit mounted on the robot is too inaccurate to be relied on to locate a robot this size. Analysis of the video feed was therefore used to complement the GPS data.

The Plan Manager is charged with keeping track of updates sent by the Progress Tracker, by both updating the XML plan, as well as displaying a refreshing graphical representation of the plan (shown in Figure 5.24). This representation uses colour to show the state of each goal, with orange representing in progress goals, blue representing non started goals, and green representing accomplished goals.

c. Results

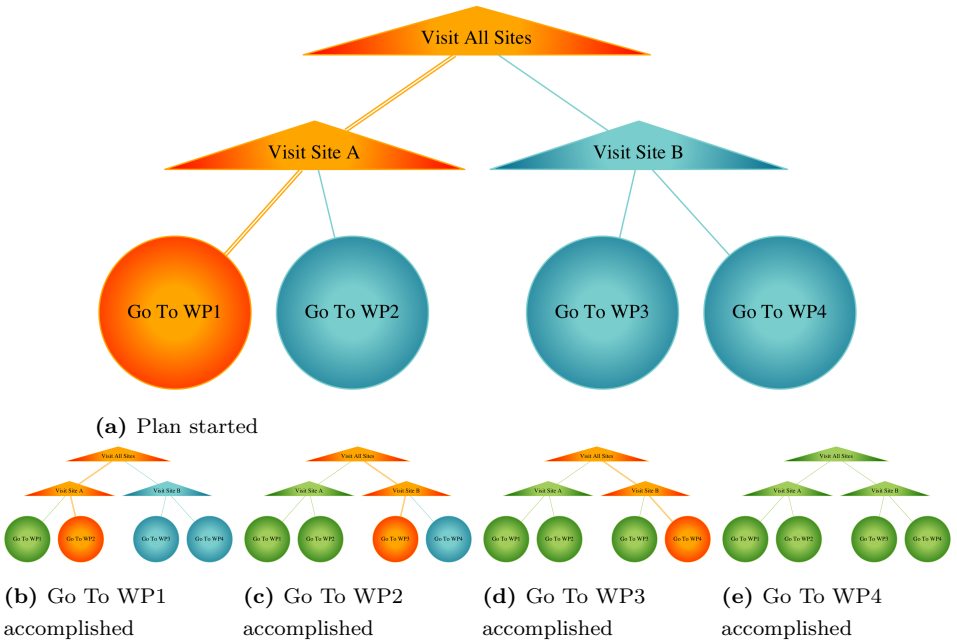
The results of the experiment are shown in Figures 5.25 and 5.26. To validate the management sub-system these results need to show that the goals accomplished by the robot have been accurately tracked, and that

```xml
<?xml version="1.0"?>
<plan-tree>
    <goal goal-type="Parent">
        <name>Visit All Sites</name>
        <status>Not Started</status>
            <subgoal goal-type="Parent">
                <name>Visit Site A</name>
                <status>Not Started</status>
                        <subgoal goal-type="Achieve">
                            <name>Go To WP1</name>
                            <status>Not Started</status>
                        </subgoal>
                        <subgoal goal-type="Achieve">
                            <name>Go To WP2</name>
                            <status>Not Started</status>
                        </subgoal>
            </subgoal>
            <subgoal goal-type="Parent">
                <name>Visit Site B</name>
                <status>Not Started</status>
                        <subgoal goal-type="Achieve">
                            <name>Go To WP3</name>
                            <status>Not Started</status>
                        </subgoal>
                        <subgoal goal-type="Achieve">
                            <name>Go To WP4</name>
                            <status>Not Started</status>
                        </subgoal>
            </subgoal>
    </goal>
</plan-tree>
```

Figure 5.22: The plan used during functional test 2.

```
# Format: "Goal Name", "Test Type 1", "Desired State 1", Confidence between 0-1, "Test Type
     n", "Desired State n", Confidence between 0-1,
# These files can contain comments
"Go To WP1", "gps", "-31.9875367009, 116.04587982, 10", "0.5", "surf", "images/WP1.jpg, 25",
     "0.5"
"Go To WP2", "gps", "-31.9876486684, 116.045953884, 10", "0.5", "surf", "images/WP2.jpg, 25",
     "0.5"
"Go To WP3", "gps", "-31.9877418368, 116.045895234, 10", "0.5", "surf", "images/WP3.jpg, 25",
     "0.5"
"Go To WP4", "gps", "-31.9875876962, 116.045832398, 10", "0.5", "surf", "images/WP4.jpg, 25",
     "0.5"
```

Figure 5.23: Goal test specifications, as used in functional test 2.



(a) Plan started

(b) Go To WP1 accomplished

(c) Go To WP2 accomplished

(d) Go To WP3 accomplished

(e) Go To WP4 accomplished

Figure 5.24: Plan progress display used during the experiment. Each subfigure shows plan state after a goal has been accomplished.

the plan structure was updated accordingly in real-time. These results are detailed below.

Progress tracking: During the experiment, the progress of the robot was tracked using two sensors in conjunction: a GPS sensor and the front-facing camera on the robot. To qualify as having accomplished each goal, the robot needed to be located near each waypoint (confirmed using GPS), and facing the target (confirmed using the SURF algorithm). As specified in Figure 5.23, both sensors need to report true before the goal is considered accomplished (confidence is 0.5 for each of them, requiring both to reach the 100% required). Figure 5.26 charts the results of the goal tests over time, as well as displaying the time at which the plan was updated with goal accomplishment information.

Of interest for progress tracking are the green and red lines, which chart distance between the robot and the waypoint in meters, and number of key point matches between the target image and what the robot's camera is pointed at, respectively. Associated to those lines are two coloured dotted

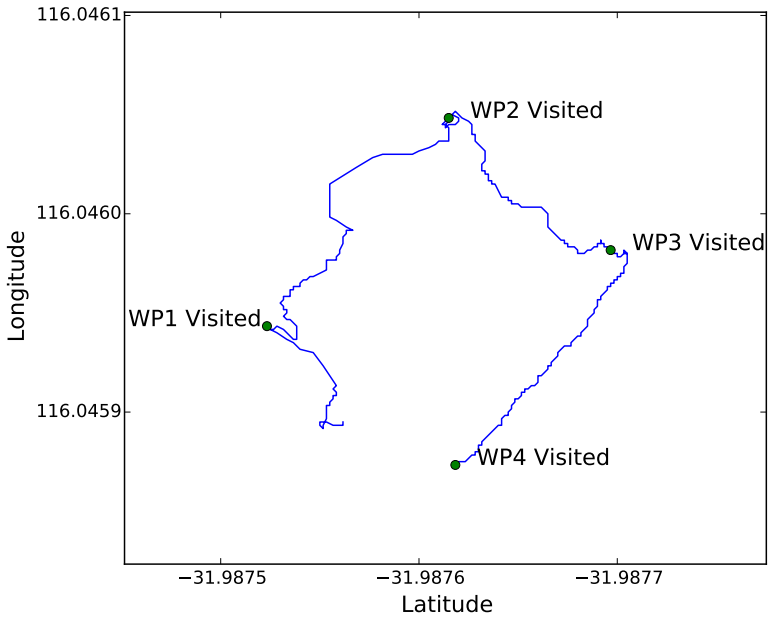


Figure 5.25: The robot's recorded GPS coordinates. The green dots indicate Goal Accomplishment occured at these coordinates.

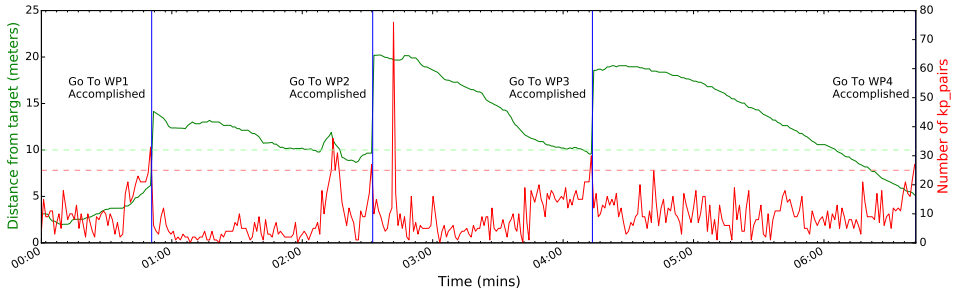


Figure 5.26: Distance from the target (green), number of key point pairs between target image and visible scene (red), as well as Goal Accomplishment (blue) over time.

lines, which mark the success thresholds for each sensor. For the GPS sensor, a value below this threshold is desired, whereas in the case of the imaging sensor, more matches equal success.

These results show that ultimately goal accomplishment is being tracked, as each time both sensors reported success, a plan update was requested. These requests reflected the state of the outside world, with the robot's state being accurately judged in each instance. These results also showcase the robustness of this approach, by demonstrating the overcoming of obstacles in the sensor data, as well as highlighting some issues with it.

The robot started quite close to waypoint 1 (WP1), as shown in Figure 5.25, and reflected in Figure 5.26, with the GPS reporting distance between robot and target as being under the threshold from the start. This means that this goal was considered to be accomplished as soon as the imaging sensor was found to be looking at the target.

"Go To WP2"s accomplishment failed to register around the 2:20 mark, with the imaging sensor returning positive, but drift in the GPS sensor caused the distance to target to be inflated. This forced the operator to try and reacquire an accurate GPS location by changing the position of the robot (temporarily losing sight of the target), before goal accomplishment could be confirmed. This is a time cost associated with the approach: once the operator has accomplished a goal, they are forced to wait and possibly have to perform additional actions for this accomplishment to be confirmed. While this time is usually relatively short (see Table 5.2), it can be increased by sensor inaccuracies (by approximately 20 seconds in this case).

"Go To WP3" shows a sensor misfire on the part of the SURF algorithm. The spike is much higher than a typical response, and occurred while the robot was still quite far from the target. The use of a second sensor ensured that goal accomplishment was not prematurely declared.

The final goal's accomplishment tracking displays a more ideal situation. The sensors reported no false positives, and the goal was declared accomplished on time. While some issues appeared and had to be overcome, the progress tracker showed that it is generally capable of tracking goal accomplishment.

Table 5.2: Time delay between goal test success and plan update, in seconds.

Goal Name	Last Goal Test Success	Plan Updated	Time to Update
Go To Wp1	10:52:48.703625	10:52:49.213235	0.509610
Go To Wp2	10:54:30.376001	10:54:30.885734	0.509733
Go To Wp3	10:56:11.450920	10:56:11.450920	0.510163
Go To Wp4	10:58:40.226899	10:58:40.735644	0.508745
Average:			0.5095627

Plan management: In comparison to the progress tracker, the plan manager has a much easier job. Simply update the plan with progress information when requested to do so. Figures 5.26 and 5.24 show that the plan manager did perform the task as requested, with Figure 5.26 showing the time at which each goal accomplishment was recorded, and Figure 5.24 displaying the resulting changes in the plan representation. It is important that these plan updates take as little time as possible to reduce operator waiting times. Table 5.2 shows the time taken by the plan manager to update the plan with progress information after being notified by the progress tracker. On average, this time was 0.5 seconds. This delay sits between the 0.1 second and the 1 second threshold of Nielsen (1993)'s response time categories, meaning that this delay is noticeable, but not long enough to disrupt the operator's train of thought. Nielsen (1993) indicates that no special feedback is required with time delays of this length.

5.5.4 Functional Test 3: Control and Autonomy Level Changes

This section describes the validation testing the system underwent before being used in the usability experiment described in Chapter 6. The goal of this validation testing was to ensure all sub-systems were functional and cooperating as expected when the system was operational. This validation was performed in two stages. Trial 1 tested manual control of the robot (as used in the Teleoperation and Assigned Responsibility "human only" mode), as well as the Management components (Progress Tracking and Plan Management). Trial 2 added the automation-based control modes of

Assigned Responsibility.

In both trials, participants (volunteers with no prior experience with the system) were asked to perform tasks required by the experiment described in more detail Section 6.2.6. This task consists of navigating, in order, to four rooms within a wooden arena, closing an open tap in each. Trial 1 only tested the "AR-H" (Assigned Responsibility - H mode only) control option, while trial 2 focused on testing the semi-automated ("AR-HA") control option, verifying that the automated switching of levels was functional. The performance of the control modes was recorded using six measures: Time taken to accomplish all of the goals, navigation errors (did the robot go to the wrong room), collisions (did the robot make contact with any of the walls), manipulation errors (was the gripper closed on empty air instead of a tap), weighted workload score (using the NASA-TLX questionnaire, see Section 6.2.5 for more details), and system usability score (using the SUS questionnaire, see Section 6.2.5 for more details).

5.5.5 Trial 1 Results

The participants controlled the robot through the entire task without automated control assistance, while the management modules were tracking progress. In both cases, the task was successfully accomplished (all goals completed). In Trial 1, inaccuracies in the robot's position tracking caused the accomplishment of the fourth navigation goal to not be detected, the goal state had to be manually overridden. These issues were fixed before Trial 2. Data collected during the trials are presented in Table 5.3.

Table 5.3: Trial 1 results.

Measures	**Operator 1**	**Operator 2**
Time Taken (mins)	13:33	27:05
Navigation Errors	0	0
Collisions	0	7
Manipulation Errors	2	7
Weighted Workload Score	35.3	57.3
System Usability Score	77.5	60

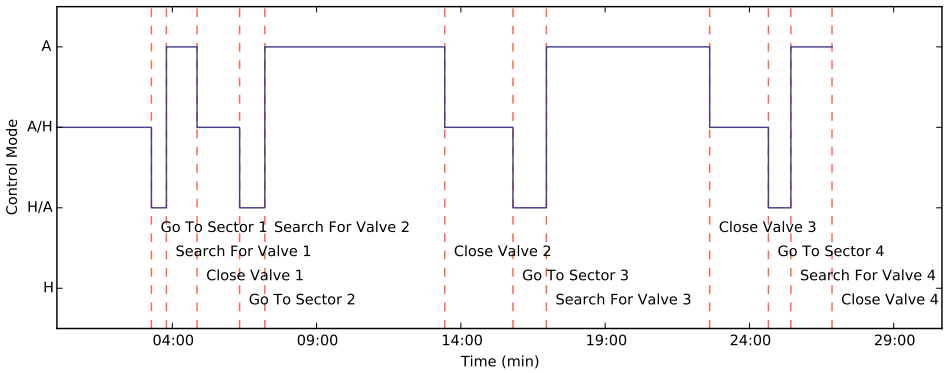


Figure 5.27: Trial 2 results. This graph charts the Level of Autonomy of the system as the task progresses.

Operator 1 was familiar with controller-based interfaces and navigating in virtual environments (through video gaming). Operator 1 had a low overall error count, while the usability score (scored using the System Usability Scale, see Appendix E) was high, and the weighted workload score (calculated using the NASA TLX Questionnaire, see Appendix D) was low, both of which are good indications.

Operator 2 was less familiar with controller-based interfaces, and played video games very infrequently. Their error count was higher than Operator 1 (14 vs. 2), and recorded higher levels of workload (57.5 vs. 35.3), as well as scoring the interface lower on the SUS questionnaire.

5.5.6 Trial 2 Results

This trial made use of the plan used for the "AR-HA" control option of the experiment described in Chapter 6. This plan requires the robot to go to four locations in turn to find and close a valve. The plan is broken down into four subgoals (close valve 1, 2, 3, 4) which are subdivided into three more sub-goals each: get to the location, find the valve, close the valve. The plan requires the level of autonomy to be changed after each goal is accomplished, with each navigation goal set to "A/H", each search goal set to "H/A", and each valve closing goal set to "A". Goal accomplishment was logged during the experiment, allowing a timeline of the level of autonomy changes during the task to be drawn up (as shown in Figure 5.27). This trial

showed all of the automation levels functioned, with the changes between level also operating as expected.

5.5.7 Implementation Validation

To achieve the aims of Assigned Responsibility, this implementation needs to fulfil the requirements outlined in Section 4.2. Table 5.4 associates the results of the functional tests presented above to the required validations steps outlined in the requirements.

Table 5.4: Requirements for Assigned Responsibility Systems

Requirement Validation	**Validation Result**
Show usability through a usability evaluation of the interface	Operators scored the interface 77.5 and 60 in the 3rd functional test (Section 5.5.4
Prove interface changes appropriately during trial runs	Successfully shown in trial 2 of the 3rd functional test
Prove interface allows human-automation cooperation during trial runs	Successfully shown in trial 2 of the 3rd functional test
Show human operators have access to a plan progress display	Plan screen shown in Figure 5.12
Prove automation capabilities during trial runs	Successfully shown in trial 2 of the 3rd functional test
Prove automation changes according to current level during trial runs	Successfully shown in trial 2 of the 3rd functional test
Prove the system supports the responsibility assignment process	Figure 5.8 shows the responsibility assignment pre-set in the plan.
Show that goal tests successfully identify goal accomplishment	Shown in the 2nd functional test (Section 5.5.3)
Show that when goal states change, the rest of system responds	Shown in the 2nd and 3rd funtional tests
Show the robot can physically perform the required tasks in trial runs	Shown in the 1st (Section 5.5.2), 2nd, and 3rd functional tests
Prove the sensors on the robot are appropriate in trial runs	Shown in the 1st, 2nd, and 3rd functional tests

5.6 Summary

This chapter presented the implementation of an Assigned Responsibility robotic teleoperation system. This implementation is based on the design presented in the previous chapter, and consists of both hardware and software components. To ensure this implementation performed as expected as well as fit the requirements outlined in Section 4.2, a series of functional tests were performed (see Section 5.5). These tests demonstrated the implementation is functional and fits those requirements, making it a working example of Assigned Responsibility as presented in the previous chapter. Thus, the implementation is demonstrably suitable for evaluating Assigned Responsibility in usability experiments.

Chapter 6

Evaluation

6.1 Overview

To evaluate the theory of Assigned Responsibility an end-to-end usability experiment was performed on the implemented teleoperation system. This experiment was designed to provide data useable to answer the research questions posed by this thesis.

The first two questions deal explicitly with the human side of teleoperation, seeking to measure concepts such as satisfaction and workload. Neither of these can be measured without volunteer operators to perform tasks using the system, and the remainder of the questions deal with the capabilities of the system while operating. As a result, the experiment was designed as a human trial of the implementation described in the previous chapter. Section 6.2 describes this experiment in detail, while Section 6.3 presents the results and analysis of these results. Finally the research questions are answered in Section 6.4.

6.2 Evaluating Assigned Responsibility

6.2.1 Overview

To test the main hypotheses of the research, a complex task (the maintenance route task) will be performed four times times by each volunteer operator, once using fully human control, and three times using

the Assigned Responsibility system in different configurations. This section describes the experiment in detail, outlining the variables, measures and procedure.

6.2.2 Tested Hypotheses

- **H1:** The explicit allocation of tasks, between robot and operator, describes an effective, efficient, and satisfying model for the teleoperation of robots.

- **H2:** The explicit allocation of tasks, between robot and operator is an effective way of reducing the cognitive load placed on the operator by traditional teleoperation systems.

- **H3:** Assigned responsibility expands the capabilities of a robot beyond those currently automatable.

6.2.3 Experimental Design

This experiment was run as a repeated measures design. This design is commonly used in user evaluations of interfaces (As shown in Table 6.1), and consists of using each participant to test all of the interfaces. This is in contrast to between subject evaluations which split participants in groups that only test one interface each. Repeated measures is efficient because it allows the use of a smaller overall number of participants (compared to the between subjects design) while not reducing the amount of data collected in the experiment. The drawback of this design is the possibility for participants to perform better in later trials as they gain experience with the system. This effect can be mitigated by ensuring participants do not undertake trials in the same order. It is important that the sample size (n) in the experiment provides for sufficient power in the inferential statistics to be used (here multiple ANOVA). Table 6.1 shows the design and sample sizes of published teleoperation interface evaluation experiments. On average, these within subjects experiments were found to use 13 subjects. This experiment used 20 participants.

Table 6.1: Number of subjects and experimental design in teleoperation user experiments.

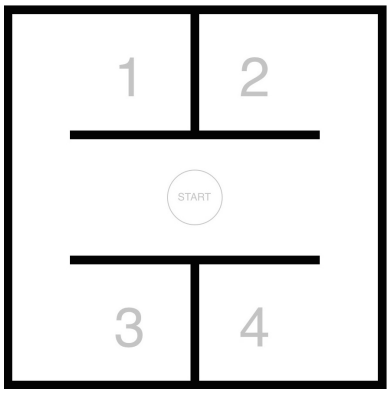
Paper	n	Design	Notes
Adams and Kaymaz-Keskinpala, 2004	30	Within Subjects	
Ciuti et al., 2012	15	Within Subjects	
Hannaford, Wood, McAffee, and Zak, 1991	5	Within Subjects	
Jaju, Banerji, and Pal, 2013	10	Within Subjects	
Mast et al., 2015	28	Between Subjects	
Olivares, Zhou, Bodenheimer, and Adams, 2003	12 and 24	Within Subjects	Experiment run twice, once with 24 novice users, once with 12 expert users.
Rosen, Hannaford, MacFarlane, and Sinanan, 1999	5 and 5	Within Subjects	Experiment run twice, once with 5 novice users (control), once with 5 expert users.
Sugimoto et al., 2005	9	Within Subjects	

The results were analysed using repeated measures analysis of variance, that allow for multiple factors to be compared over the different treatments, while accounting for the within subject nature of the experiment.

6.2.4 The Task

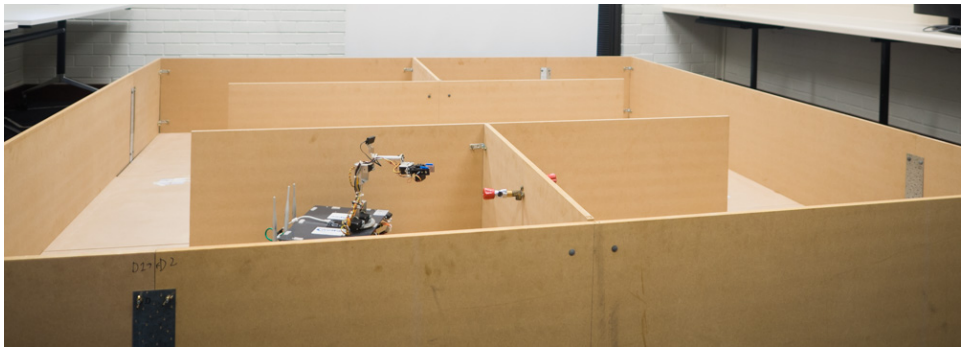
The task in this experiment consists of a mockup maintenance route suggestive of an industrial context. This task needed to be complex enough to properly test the human-machine system, yet not so complex as to be too difficult and time consuming for the human operators to learn, and for the automation to accomplish[1]. The robot must navigate to four sectors, each containing a valve that must be closed using the robot's arm

[1]This balance was achieved by testing the task with human operators and the automation with the following criteria in mind: the task needed be accomplishable by both, but not be so easy as to be accomplishable without errors being recorded. At the same time, proficiency in the task needed to be reachable in a reasonable amount of time so an operator could be trained and perform the experiment in a single 2 to 3 hour session. The results shown in the initial trials (see Section 5.5.4) showed the task fit those requirements.



(a) Map of the setup.

(b) The operator cannot see the robot directly.



(c) Photograph of the experimental setup.

Figure 6.1: Experimental setup. The robot must be guided through all four sectors, to close all four valves.

and gripper. Figure 6.1 shows the map and a photograph of the experimental setup. The task is broken down into four subgoals: Close Valve 1, 2, 3, and 4 (The closing order will change every run, and be provided to the operator before each run). Each of these subgoals is subdivided into three more subgoals, which require the operator to navigate to the sector, find the valve, and close the valve. When all of the valves have been closed, in the correct order, the task is accomplished.

To compare Assigned Responsibility to full human control, the experiment must test both control paradigms. This requires the robot to be operated using multiple control options throughout each volunteer's participation. These control modes cover teleoperation as well as a range of configurations of the Assigned Responsibility interface.

6.2.5 Measures

The following measures were collected during the experiment:

- Objective measures:
  - Effectiveness: Was the task accomplished? If not, to what extent was it accomplished? This will be measured by counting the number of taps closed at the end of the run. A time limit of ten minutes is imposed on the participants for each valve closing subgoal. If the time limit runs out, they are asked to proceed with the next valve.
  - Efficiency: How well was the task accomplished? Time taken (seconds) and non-critical mistakes (Errors that don't cause the task to fail completely, such as collisions or localisation errors. Critical mistakes count against Effectiveness instead, as they prevent the task from being completed).
- Qualitative Measures
  - Operator Workload: How much mental strain was the operator under? This is measured using the NASA Task Load Index (TLX) (Hart & Staveland, 1988). Three sets of Weighted Workload (WWL) scores (one for each run) and one set of weights will have been collected during the course of the experiment. The individual WWL scores summarise the subjective workload the operator is under.
  - Usability/ Satisfaction: How easy to use is the system? This is measured using the System Usability Scale (SUS) (Brooke, 1996). The effort and frustration components of the TLX can also provide additional information about this.

6.2.6 Experimental Protocol

1. The participant is welcomed and introduced to the robot, the task, and the setting. The experimental procedure is explained and the

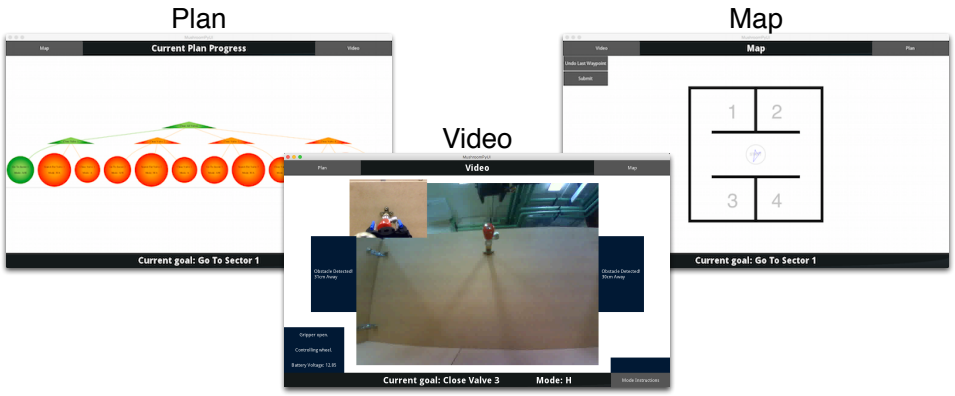


Figure 6.2: Screens available during the Assigned Responsibility runs.

participant is allowed to practice controlling the robot with each of the control modes.

2. The participant is allowed to read the NASA TLX instructions to familiarise themselves with the measured factors before they undertake the tasks.

3. The participant is allocated each of the control option T, AR-H, AR-HA, or AR-A in a random sequence.

    - AR-H: Assigned Responsibility with all subgoals set to human control (mode H)

      In this control option, the operator has access to all of the features of the Assigned Responsibility interface, as shown in Figure 6.2. None of the subgoals are set to use any automation however, and the operator is therefore responsible for all control actions. If the time limit in any sector is exceeded, the operator is asked to press the "Goal Failed" button, which will set that sector's goals to failed, and make the next sector's first goal the current goal.

    - AR-HA: Assigned Responsibility with subgoals set to a range of modes (H, H/A, A/H, or A)

      In this control option, the operator also has access to all three screens shown in Figure 6.2, but is accomplishing tasks in cooperation with varying levels of automation. If the time limit

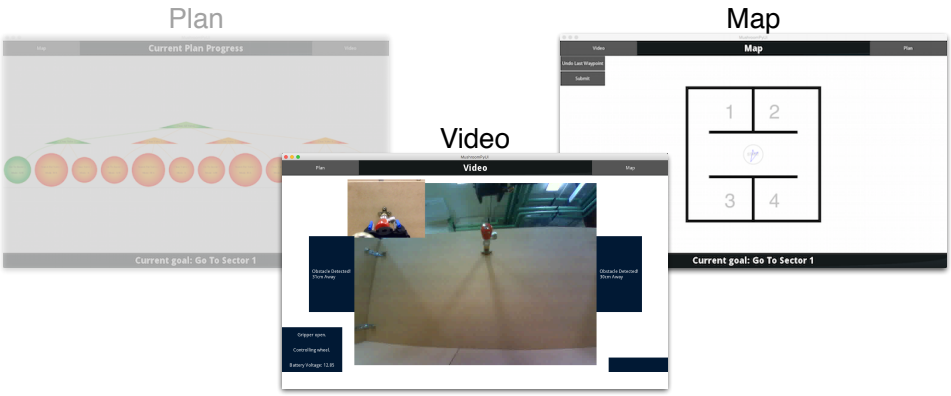


Figure 6.3: Screen subset available during the Teleoperation run.

in any sector is exceeded, the operator is asked to press the "Goal Failed" button, which will set that sector's goals to failed, and make the next sector's first goal the current goal.

- AR-A: Assigned Responsibility with all subgoals set to full automation (mode A)

  In this control option, the operator also has access to all three screens shown in Figure 6.2, but is only acting as an observer. The automation will perform the task with no input from the operator. If the time limit in any sector is exceeded, the operator is asked to press the "Goal Failed" button, which will set that sector's goals to failed, and make the next sector's first goal the current goal.

- T: The full human control (referred to during the experiment as teleoperation) interface.

  In this control option, the operator is asked to accomplish the task without automated aids, or support from Assigned Responsibility components such as Progress Tracking and Plan Management. The order the sectors are to be visited in is given to the operator on paper. The operator only has access to the main Video screen and the Map screen as shown in Figure 6.3. If the time limit in any sector is exceeded, the operator is asked to proceed to the next listed sector and accomplish the required tasks there.

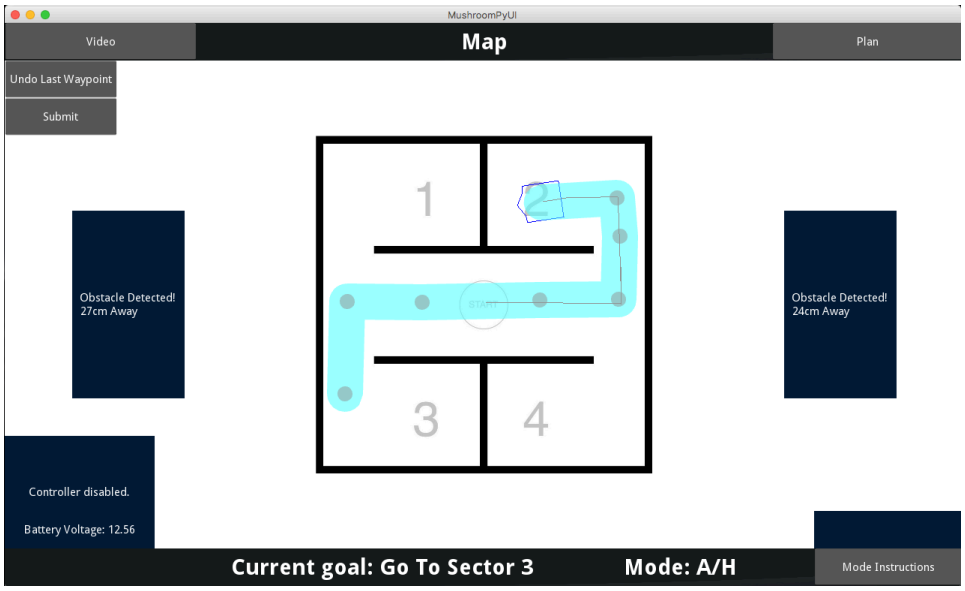


Figure 6.4: Operator setting waypoints using the map in the AR-HA control option.

4. The participant undertakes the task with the allocated control option. The task consists of three goals repeated in each of four sectors. There is a ten minute time limit on each of these repetitions. If a participant exceeds the time limit, they are asked to proceed with the next set of goals.

   4.1 Go to sector:

   The robot needs to navigate from its current location to the specified sector (1, 2, 3, or 4).

   - AR-H: The goal is set to H mode (full human control). The operator must use the controller to guide the robot's movements. The map screen can be used to review current position as well as the target location.
   - AR-HA: The goal is set to A/H Mode (Automated with human supervision). The operator is asked to switch to the map screen and input waypoints on the map to guide the robot to the desired location (As shown in Figure 6.4). Once the operator submits the waypoints, the automation executes the required moves.

- AR-A: No action required.
- T: In teleoperated control, the operator has manual control over the robot's actions, and must confirm the results of these actions themselves as there is no automated progress tracking.

4.2 Find Valve:

This goal is accomplished when the arm-mounted camera is properly facing the valve (as detected by the colour tracking algorithm described in Section 5.4.3). The robot must be used to search the sector until this valve is found.

- AR-H: The goal is set to H mode (Full human control). The operator must use the controller to guide the robot's movements.
- AR-HA: The goal is set to H/A mode (Human control, Automation assisting). The operator must use the controller to guide the robot's movements, the automation will provide an overlay highlighting the valve's location in the camera feed if detected.
- AR-A: No action required.
- T: In teleoperated control, the operator has manual control over the robot's actions, and must confirm the results of these actions themselves as there is no automated progress tracking.

4.3 Close Valve:

The goal is accomplished when the valve is closed. The robot's arm must be used to reach for and turn the valve in order to close it. In the AR modes, the system checks this and updates the current goal when the tap is closed. In T, the operator must visually confirm the accomplishment of the task (i.e. by turning until the gripper isn't moving anymore).

- AR-H: The goal is set to H mode (fully human control). The operator must use the controller to guide the robot's movements (Shown in Figure 6.5).

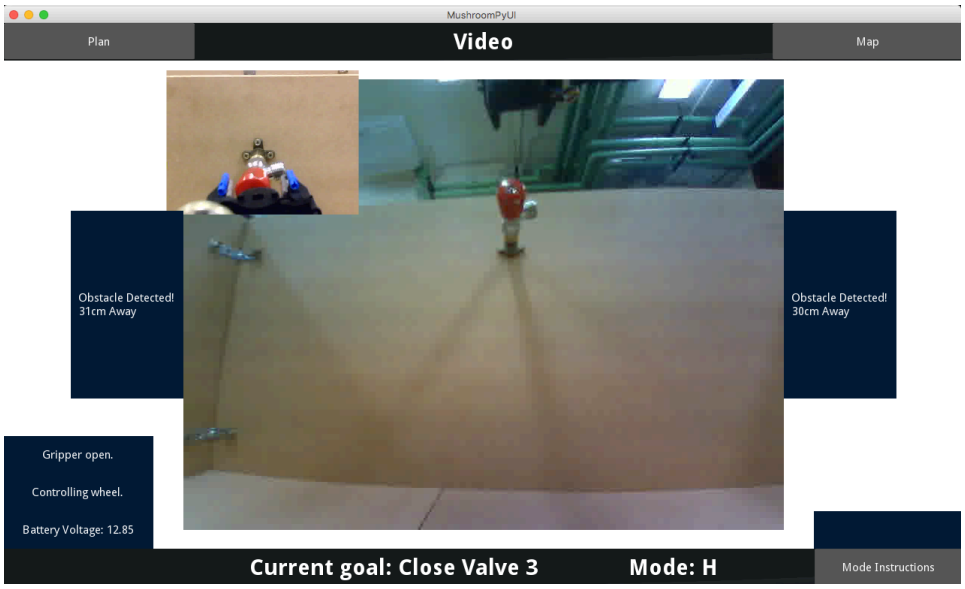


Figure 6.5: Operator closing the valve in H mode. Inset shows the view from the arm camera.

- AR-HA: The goal is set to A mode (fully automated control). The operator does not have control over any of the robot's actions. In case of repeated failure from the automation, the operator will be asked to take control and finish the task.
- AR-A: No action required.
- T: In teleoperated control, the operator has manual control over the robot's actions, and must confirm the results of these actions themselves as there is no automated progress tracking.

5. Objective data is recorded about the task's execution, namely: Effectiveness (Success rate) and Efficiency (Time taken, mistakes made).

6. After undertaking each task, regardless of outcome, the participant will be asked to complete the ratings section of the NASA-TLX questionnaire (see Appendix D).

7. After all three control options have been used, the participant will be asked to complete the weightings section of the NASA-TLX

questionnaire (see Appendix D), as well as the SUS questionnaire (see Appendix E)

6.2.7 Limitations

- **Operator Selection:** The operators were selected by convenience, possibly limiting the usefulness of generalisations drawn from this experiment. This was a practical necessity, since a pool of robot operators from which a more representative sample could be drawn was not available.

- **Implementation Testing:** While it does tackle the broader hypotheses listed above, this experiment will be testing only one possible implementation of each mode. More experience with the use of practical implications would be needed to be sure of every aspect of its value.

6.3 Results

6.3.1 Overview

This section presents the results of the experiment. These results have been split into four categories: System Usability Scale (SUS) results, NASA Task Load Index (TLX) results, objective Measures, and Informal Survey results. For each category, the relevant statistics for the dataset are presented, and analysed to extract meaningful results.

6.3.2 System Usability Scales

Each System Usability Scale (SUS) questionnaire reports a single usability number (scored out of a hundred, the higher the number, the better). This score cannot be used as a percentage however, and is only useful as a way of comparing an interface to others rated using the SUS. Bangor, Kortum, and Miller (2009) provide a large dataset of these scores, separated by interface type. Interfaces should strive to rate higher than the average in

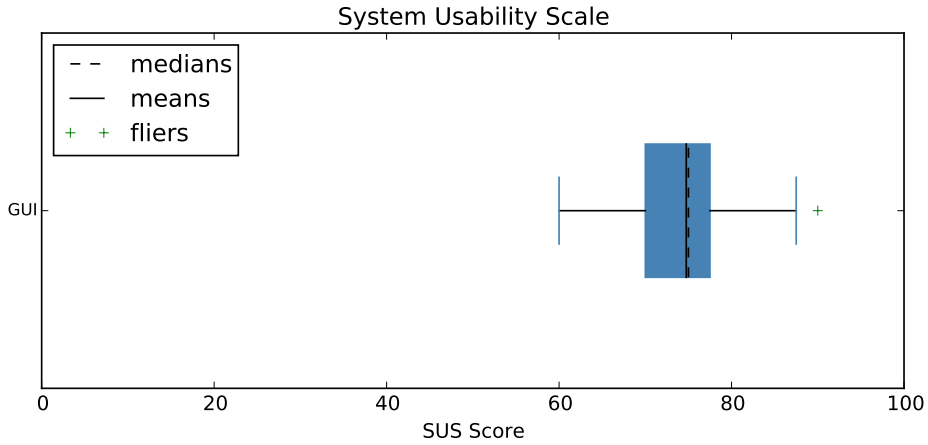


Figure 6.6: System Usability Scale results for the interface.

their category. Outside of use as a comparison tool, the scores can also be used to classify interfaces into four categories (Bangor et al., 2009):

- Under 70: Usability issues likely present.
- 70 to under 80: Acceptable usability.
- 80 to under 90: Good usability.
- 90 to 100: Exceptional usability.

The results of the questionnaire are presented in Figure 6.6. The mean score for the interface is **74.75**, which places it in the acceptable usability segment of the scale. The closest interface type provided by Bangor et al. (2009) is GUI, which have an average score of 76.2, making the system's interface acceptable, but slightly worse than an average GUI.

6.3.3 Weighted Workload

The weighted workload score returned by the TLX consist of two parts. The first is the raw scores for each scale (mental demand, physical demand, temporal demand, performance, effort, and frustration), as recorded by the participants for each control option. The second part is the weighted workload score (WWL) which combines the raw values into a

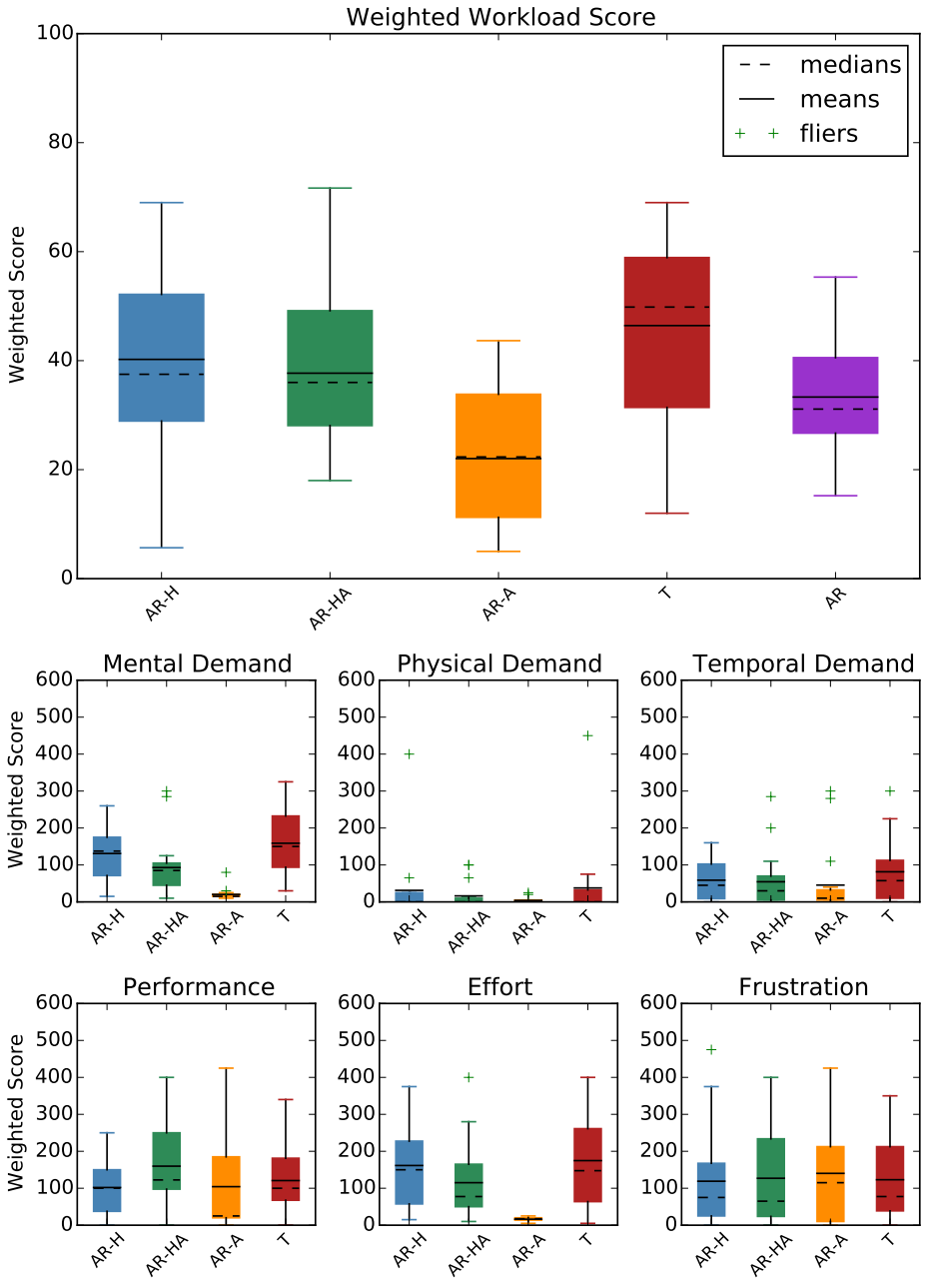


Figure 6.7: NASA Task Load Index results. The Weighted Workload score plot shows the weighted scores for each control option (AR-H, AR-HA, AR-A, T) and for the Assigned Responsibility modes averaged (AR). The smaller plots show the weighted scores given by the participants to each control option for each measure. A higher score means more workload.

Table 6.2: Definitions for the acronyms present in the results tables (From (Lawrence, 2013)).

Acronym	Definition
DFn	Degrees of Freedom in the numerator (a.k.a. DFeffect)
DFd	Degrees of Freedom in the denominator (a.k.a. DFerror)
SSn	Sum of Squares in the numerator (a.k.a. SSeffect)
SSd	Sum of Squares in the denominator (a.k.a. SSerror)
F	F-value
p	p-value (probability of the data given the null hypothesis)
p<.05	Highlights p-values less than the traditional alpha level of .05
ges	Generalized Eta-Squared measure of effect size
GGe	Greenhouse-Geisser epsilon
p[GGe]	p-value after correction using Greenhouse-Geisser epsilon
p[GGe]<.05	Highlights p-values (after correction using Greenhouse-Geisser epsilon) less than the traditional alpha level of .05
HFe	Huynh-Feldt epsilon
p[HFe]	p-value after correction using Huynh-Feldt epsilon
p[HFe]<.05	Highlights p-values (after correction using Huynh-Feldt epsilon) less than the traditional alpha level of .05
W	Mauchly's W statistic

single score using weights assigned by the participants to each scale (shown in Figure 6.7). This way, the WWL score accounts for the expected variations between individuals' subjective experience of the workload, making WWL scores comparable across participants. The individual weighted scores are also included in Figure 6.7.

As the experiment is a within subjects design, the suitable statistical test is a one-factor repeated measures analysis of variance (ANOVA), the factor being the control options. The ANOVA tests the null hypothesis, which in this case is that the factor has no significant effect on the measures. Before the ANOVA results can be trusted however, the data set has to be tested for sphericity. The ANOVA calculations assume the data are spherical, that is to say that the variations between factors are relatively homogenous (This only matters when comparing more than two control options). If the data are not spherical, the confidence value returned by the ANOVA must be corrected.

To begin, the Assigned Responsibility options as a whole (AR) are compared to the direct teleoperation option (T). The AR scores were calculated by averaging the scores given by each operator to each of the

Table 6.3: ANOVA results for the Weighted Workload Scores of T and AR. Acronyms explained in Table 6.2.

DFn	DFd	SSn	SSd	F	p	p<.05	ges
1	19	1719.0123	1828.5185	17.8621	4.570×10^{-4}	*	0.2002694

AR modes, and are shown in Figure 6.7. Comparing these data to the T data does not require sphericity calculations as there are only two data sets being compared. The results of this analysis are presented in Table 6.3.

These results show that there is a significant difference between these two factors, and Figure 6.7 shows that the mean of AR is lower than the mean of T. As a whole then, the WWL score of Assigned Responsibility is significantly lower than the WWL of the direct teleoperation option.

To gain more insight into the differences between AR and T, another ANOVA is performed, comparing each control option to each other. The results of this ANOVA are presented in Table 6.4. Table 6.4a, shows the results of the ANOVA calculations, Table 6.4b shows the results of the sphericity test, and Table 6.4c displays the possible sphericity corrections. These results show that the data are spherical, and that the null hypothesis of the ANOVA can be rejected ($p < 0.05$). At least one of the factors is significantly different to the others.

This result does not indicate which factors are different, so a post-hoc analysis must be performed to obtain this information. Table 6.5 shows the results of the pairwise comparison done to compare the factors.

Table 6.4: ANOVA results for the Weighted Workload Scores of each control option. Acronyms explained in Table 6.2.

(a) ANOVA Results

DFn	DFd	SSn	SSd	F	p	p<.05	ges
3	57	6465.8667	10370.0222	11.8468	3.900×10^{-6}	*	0.2597953

(b) Mauchly's test for sphericity.

W	p	p<.05
0.3811919	0.004392525	*

(c) Sphericity corrections.

GGe	p[GG]	p[GG]<.05
0.6123966	1.710×10^{-4}	*

HFe	p[HF]	p[HF]<.05
0.6747885	9.262×10^{-5}	*

Using a confidence level of 0.05, this table indicates that the AR-A control option is significantly different to the other three control options, which are not significantly different to each other. More information can be gained by comparing the means of the WWL scores presented in Figure 6.7. T has a higher mean than AR-H, which in turn has a higher mean than AR-HA, which in turn has a higher mean than AR-A. While this is the case, only AR-A is significantly different to the others, which may limit the possible claims coming from this result.

Table 6.5: Pairwise comparisons using paired t-tests.

	AR.A	AR.H	AR.HA
AR.H	0.015676571		
AR.HA	0.000161499	1.00000000	
T	0.000147786	0.28314586	0.4295323

6.3.4 Objective Measures

a. Overview

The previous section compared the WWL scores of the control options to each other, and of the AR options as a whole compared to T, using ANOVAs. This section is performing the same analysis with three different dependent measures recorded during the experiment: errors, time, and goal failures. Errors is composed of three counts: collisions (how many times the robot collided with a wall), navigation errors (how many times the robot entered the wrong sector), and manipulation errors (how many times the robot's gripper closed on thin air when going to grasp the valve). Time is the number of seconds taken to accomplish the task, and goal failures is a count of the number of times a valve was not closed. The analysis process is the same as was described above, so for brevity, the details will not be repeated.

b. Errors Made

The error data collected are presented in Figure 6.8. When T is compared to the grouped AR options, operations using T were found to result in significantly lower error counts than those performed with AR options (as

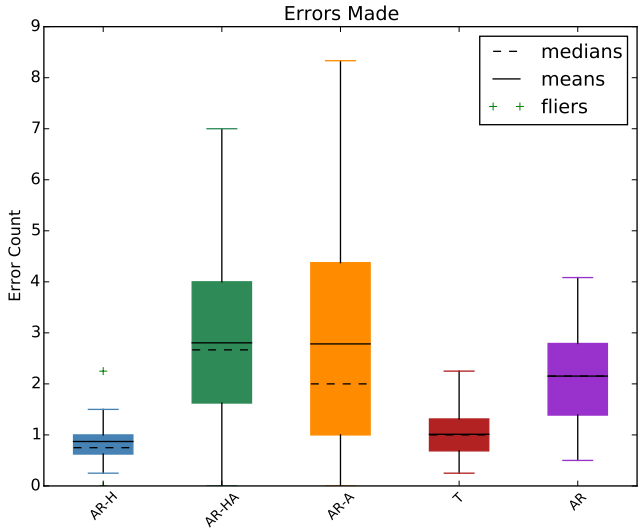


Figure 6.8: Combined errors made per tap closed. Errors are collisions, navigation errors, and manipulation errors.

Table 6.6: ANOVA results for the error counts of T and AR. Acronyms explained in Table 6.2.

DFn	DFd	SSn	SSd	F	p	p<.05	ges
1	19	13.0023	11.2149	22.02828	1.583×10^{-4}	*	0.3367984

shown in Table 6.6).

The ANOVA results for the option to option comparison are shown in Table 6.7, and the pairwise comparisons in Table 6.8.

The ANOVA results show the data is spherical, and that the null hypothesis is rejected, there is a significant difference between at least one factor and the others at the 0.05 level. The post-hoc analysis shown in Table 6.8 show that T and AR-H are not significantly different to each other and that the same applies to AR-HA and AR-A. These two groups are, however, significantly different to each other. Figure 6.8 shows that the means of AR-HA and AR-A are higher than those of T and AR-H. We can conclude that there were significantly more errors made while AR-A and AR-HA were being used than when T and AR-H were being used.

Table 6.7: ANOVA results for the errors made with each control option. Acronyms explained in Table 6.2.

(a) ANOVA Results

DFn	DFd	SSn	SSd	F	p	p<.05	ges
3	57	68.8093	111.05356	11.7725	4.170×10^{-6}	*	0.3018191

(b) Mauchly's test for sphericity.

W	p	p<.05
0.20718776	3.955×10^{-5}	*

(c) Sphericity corrections.

GGe	p[GG]	p[GG]<.05
0.6642785	1.075×10^{-4}	*

HFe	p[HF]	p[HF]<.05
0.7419757	5.046×10^{-5}	*

Table 6.8: Pairwise comparisons using paired t-tests.

	AR.A	AR.H	AR.HA
AR.H	0.009349159		
AR.HA	1.000000000	0.0014837086	
T	0.017775685	1.0000000000	0.0004448004

c. Time

The time data is presented in Figure 6.9. When T is compared to the grouped AR options, tasks undertaken using T were found to take significantly more time than those performed with AR options (As shown in Table 6.9).

The ANOVA results for the option to option comparison are shown in Table 6.10, and the pairwise comparisons in Table 6.11. The ANOVA results show that the null hypothesis is rejected, however Table 6.10b shows that the data is not spherical, and that corrections must be applied before the results of the ANOVA can stand. Table 6.10c shows that when either correction is applied, the confidence level is still below the threshold of 0.05, allowing us to conclude that the null hypothesis can be safely rejected.

Table 6.9: ANOVA results for the time taken by T and AR. Acronyms explained in Table 6.2.

DFn	DFd	SSn	SSd	F	p	p<.05	ges
1	19	69476.8556	25166.8396	52.4524	7.108×10^{-7}	*	0.3832017

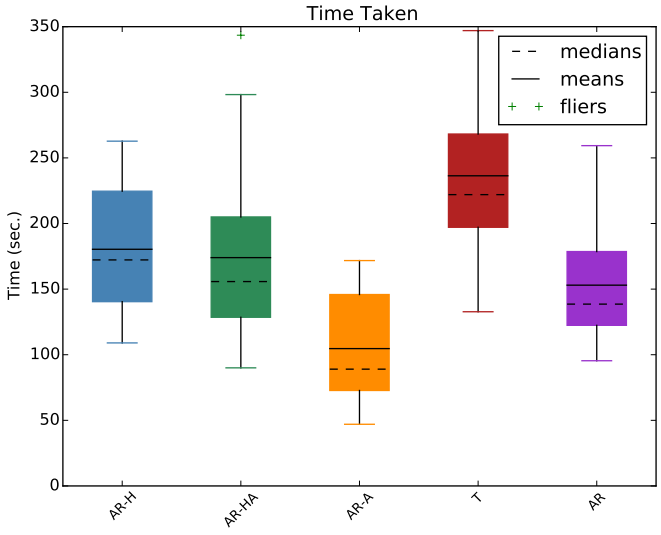


Figure 6.9: Time taken per tap closed.

The pairwise comparisons shown in Table 6.11 show that only AR-H and AR-HA are not significantly different. When viewed with the data shown in Figure 6.9, this shows us that T is significantly slower than the other options, and that AR-H and AR-HA are significantly slower than AR-A.

d. Goal Failures

The goal failures collected are presented in Figure 6.10. When T is compared to the grouped AR options, tasks undertaken using T were found to fail goals significantly less than those performed with AR options (as shown in Table 6.12).

The ANOVA results for the option to option comparison are shown in Table 6.13, and the pairwise comparisons in Table 6.14. The ANOVA results show the data is spherical, and that the null hypothesis is rejected, there is a significant difference between at least one factor and the others. The post-hoc analysis shown in Table 6.14 shows that T and AR-H are not significantly different, and that neither are AR-HA and AR-A. However these two groups are significantly different, with T and AR-H failing to complete significantly less goals than AR-HA and AR-A. Some caution is warranted here, since the observed rates are very low in all conditions.

Table 6.10: ANOVA results for time taken with each control option. Acronyms explained in Table 6.2.

(a) ANOVA Results

DFn	DFd	SSn	SSd	F	p	p<.05	ges
3	57	174731.6281	93211.9517	35.6167	4.239×10^{-13}	*	0.4194909

(b) Mauchly's test for sphericity.

W	p	p<.05
0.8270385	0.6442822	

(c) Sphericity corrections.

GGe	p[GG]	p[GG]<.05
0.8962289	5.810×10^{-12}	*

HFe	p[HF]	p[HF]<.05
1.05803327	4.239×10^{-13}	*

Table 6.11: Pairwise comparisons using paired t-tests.

	AR.A	AR.H	AR.HA
AR.H	0.0000076398		
AR.HA	0.0000357833	1.000000000	
T	0.0000000420	0.006090529	0.0004670786

Table 6.12: ANOVA results for the time taken by T and AR. Acronyms explained in Table 6.2.

DFn	DFd	SSn	SSd	F	p	p<.05	ges
1	19	2.17778	1.9333	21.4023	1.842×10^{-4}	*	0.5297297

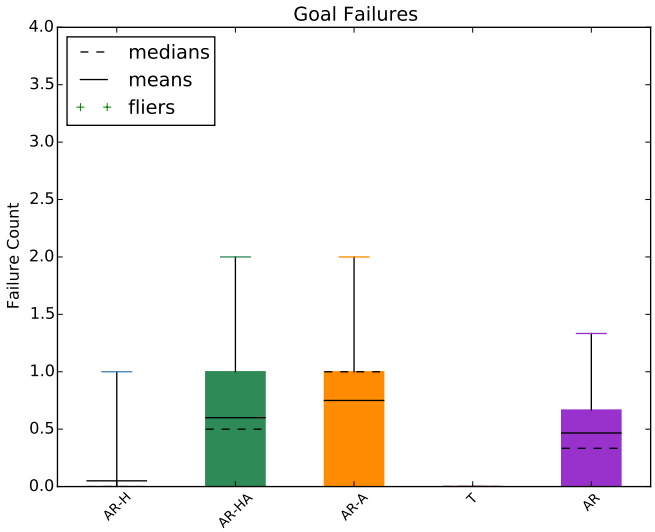


Figure 6.10: Goal failures recorded.

Table 6.13: ANOVA results of goal failures of each control option. Acronyms explained in Table 6.2.

(a) ANOVA Results

DFn	DFd	SSn	SSd	F	p	p<.05	ges
3	57	8.7	10.8	15.3056	2.022×10^{-7}	*	0.3085106

(b) Mauchly's test for sphericity.

W	p	p<.05
0.18352409	1.507×10^{-5}	*

(c) Sphericity Corrections

GGe	p[GG]	p[GG]<.05
0.5511766	5.757×10^{-5}	*

HFe	p[HF]	p[HF]<.05
0.597059	3.215×10^{-5}	*

Table 6.14: Pairwise comparisons using paired t-tests.

	AR.A	AR.H	AR.HA
AR.H	0.0024677632		
AR.HA	1.0000000000	0.011880701	
T	0.0009738713	1.000000000	0.005239164

6.3.5 Informal Survey Results

Participants were asked three questions after completing the experiment and the TLX and SUS questionnaire: i) If they considered themselves to be familiar with video game controllers, ii) Which control option they enjoyed the most, and iii) To rank the remaining control options from most enjoyable to least enjoyable. As this survey is not a properly vetted questionnaire, it cannot be used as a way of quantifying participant impressions. However, these questions allow participants to self-report their impressions in a manner more specific to this experiment. This means this data can be used to provide insight into why participants preferred an option over another for example.

To the first question, all twenty participants replied they had experience with video game controllers. To the second question, thirteen participants said they enjoyed using AR-H the most, five preferred AR-HA, one had AR-A at the top of their list, and one ranked T highest.

To process the results of the third question, a score was assigned to each control option based on how they were ranked by each participant. Four

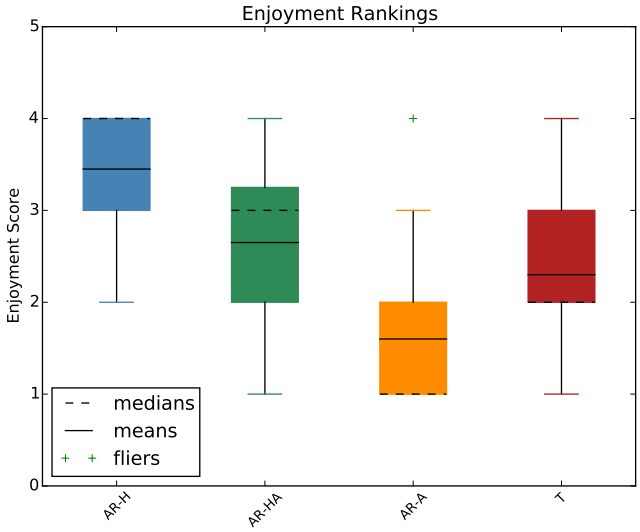


Figure 6.11: Participant ranking of each control option by enjoyment.

points were given to the highest ranked, three to the second highest, and so on. The distribution of scores assigned to each option are are presented in Figure 6.11. Overall, AR-H scored 69 points, AR-HA 53, AR-A 32, and T 46.

6.4 Discussion

6.4.1 Overview

This experiment was designed to provide data useful for answering the research questions posed by this thesis. This section will use the data provided above to answer these questions, as well as draw additional conclusions.

6.4.2 Is Assigned Responsibility an effective, efficient, and satisfying model for the teleoperation of robots?

To answer this question, the experiment pitted the Assigned Responsibility (AR) options against the more traditional direct teleoperation (T) option. The measures of interest here are the objective measures (time, errors, and

goal failures) to test effectiveness and efficiency, and the informal enjoyment question and SUS questionnaire for satisfaction. To answer this question in the affirmative, the data would have to show that Assigned Responsibility is at least as good as direct teleoperation.

a. Effectiveness

An effective system is a system that accomplishes the tasks set to it. The goal failure data presented in Section 6.3.4 serves as a suitable measure of effectiveness as it showcases wether or not a task set to the system was accomplished.

The goal failure data showed that the T and AR-H options caused significantly less goal failures than the AR-HA and AR-A options. In fact there were no goal failures when T was employed, and just the one when AR-H was employed. This goal failure recorded when AR-H was in use was an operator error: an operator forgot to open his gripper after closing a tap, and undid that tap when returning the wrist to a central position. Out of 80 possible such failures, T recorded a 0% failure rate, and AR-H a 1.25% failure rate.

The goal failures recorded during the other two options were caused by the automated sections of the tasks, with a large portion being caused by mapping inaccuracies accumulating and causing the robot to become lost during navigation tasks. AR-HA recorded a 15% failure rate, and AR-A a slightly higher 18.75% failure rate. This clearly shows the vulnerability of an AR system to its weakest automated task, and suggests that either error recovery strategies be implemented to recover from these failures (which wasn't the case here), or that the level of autonomy of the goal be brought back down until the automation is improved.

The overall comparison between AR and T shows that the AR options together caused significantly more goal failures, and as stated above this difference is caused by the introduction of automation into the control loop. Altogether, the AR options recorded a 11.67% failure rate.

These data show that out of all the recorded runs, the AR options were less effective than the T option in this case. Assigned Responsibility can

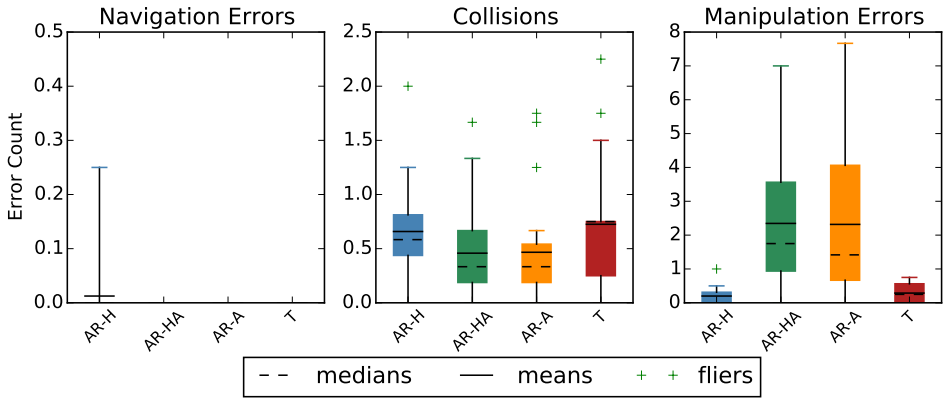


Figure 6.12: Errors made per tap closed, by type.

however operate very similarly to direct teleoperation if needed (as was done with the AR-H option) and therefore potentially achieve similar effectiveness levels. This does diminish some of the benefits of Assigned Responsibility, but not all, as will be shown later. It is important to note that this experiment can only test this particular implementation of automation, which is not claimed to be indicative of the most up-to-date automation techniques.

b. Efficiency

Efficiency was measured during this experiment as a combination of two factors: non-fatal errors (fatal errors being errors that caused goal failures), and time. These non-fatal errors (collisions, manipulation errors, and navigation errors), do not cause the task to fail, but serve as a useful measure of control quality. An efficient use of the system should result in a low error, low time task accomplishment.

Section 6.3.4 presented the error data collected during the experiment. The data showed that the use of T and AR-H resulted in significantly fewer errors than AR-HA and AR-A. Some insight into these errors can be gained by breaking down the error counts into their three components.

As shown in Figure 6.12, navigation errors were extremely rare, occurring only once during the experiment. Collisions were present in all modes, but occurred more frequently in the manual control modes (AR-H

and T). Manipulation errors were far more frequent in the modes that used automation. Both AR-HA and AR-A used the same algorithm to close the tap, so their similar scores are to be expected. While the automated modes made more errors there, they were also capable of making closure attempts at a much faster rate than a human operator, resulting in a higher error count, but a faster task accomplishment rate as discussed below.

The time data showed that T was significantly slower than AR-H and AR-HA, which were significantly slower than AR-A. This information highlights important differences in all of the modes. In AR-A, being fully automated, the system never had to wait for operator input, which had two consequences: a faster task accomplishment rate, and a higher goal failure rate. In AR-HA, the amount of time the robot spent moving would probably have been fairly similar to that of AR-A, however the automation spent time waiting for waypoints to be set by the human operator, which delayed it.

One of the more surprising differences lies with AR-H and T. With both those options, all of the control task was performed by the human operator. This is reflected in the similar error rates for both options. However, AR-H had a significantly faster accomplishment rate. This difference comes almost entirely from the goal accomplishment tracking during the AR-H runs. With the T option, the operator had to confirm the closure of the tap visually by testing it repeatedly until satisfied. In the AR-H runs, the goal accomplishment tracking was monitoring the strain placed on the wrist by the tap, allowing for a fast detection of the tap's state, cutting down the number of grasp attempts needed to close the tap, and thus saving time.

These two sets of data tell a different story, with T outclassing AR as a whole on errors made, but causing a significant increase in time taken for task accomplishment. Overall, the Assigned Responsibility system resulted in faster task accomplishment across the board, with the ability to trade a higher error rate for more speed or vice versa (by switching from human control to (semi-automated control) if needed, making it more efficient than the direct teleoperation option.

c. Satisfaction

To measure satisfaction, two data sets were collected: the System Usability Scale (SUS), and the informal survey. The SUS provided information on the participants' opinion of the interface and ease of use of the system. While the SUS score does not distinguish between control mode options, it provides a view of the overall system, which was operating in Assigned Responsibility options during the majority of the experiment. The SUS questionnaire reported acceptable usability, providing evidence that poor usability did not obscure the test results, at least.

The informal survey collected the participants' opinion of each control option, having them rank the options based on how enjoyable they found them. The most enjoyed option was AR-H, with over half of the participants ranking it highest. Participants commented that they enjoyed being in control of the robot, but found the goal accomplishment tracking extremely useful, saving them time and effort when closing the tap. The second most enjoyed option was AR-HA, with those participants citing that this option relieved them of work and was the easiest to use. AR-A and T were both only preferred by one participant each, AR-A because the participant found it required the least work and succeeded in the task, and T because the participant found the automation too unreliable, and would have rather checked the success of tasks manually than to rely on the goal accomplishment tracking.

When the rest of the rankings are taken into account, AR-H still leads the way, followed by AR-HA, followed by T, and with AR-A at the bottom. AR-A was the least enjoyed, with the majority of the participants enjoying being in control, some finding watching the automation struggle a frustrating experience, and finally some stating they felt it was a waste of their time to be there when they couldn't intervene. This observation is not surprising as it supports one of the basic principles of user interface design as proposed by Shneiderman, Plaisant, Cohen, and Jacobs (2009), "support internal locus of control", which states that users of a system dislike not feeling in control of that system.

Collectively, the Assigned Responsibility options were reported by the

participants to be more enjoyable than direct teleoperation.

6.4.3 Is the explicit allocation of tasks, between robot and operator, an effective way of reducing the workload placed on the operator by traditional teleoperation systems?

Section 6.3.3 presented the results of the TLX questionnaire, and found that when compared option to option, the only control option that was singled out as providing significantly lower workload on the operator is AR-A. Interestingly however, even though AR-A had a lower Weighted Workload (WWL) score than the other options, participants still reported some workload even though they were simply asked to observe the robot perform the task unassisted. Figure 6.7, shows the individual components of the WWL, which provide some insight into the workload felt by participants with AR-A. Unsurprisingly, mental demand, physical demand, and effort are all very low, however frustration, performance, and temporal demand are all relatively high (close to those felt by the other options).

While AR-H and AR-HA were not individually significantly different to T, they scored slightly lower on average, and the grouped AR options rated significantly lower than T. For this question therefore, we may say that real improvements in operator workload are possible, but appear to become important only when the operator is maximally relieved of effort by full automation.

Using work by Grier (2015), it is possible to compare the WWL scores of the modes to the WWL scores of over a thousand systems (from a wide variety of areas, including robot control systems), and gain an understanding of how hard the operators were working during the experiment. Over all systems, Grier (2015) reported a mean WWL score of 48.74, higher than the mean of all four control options (AR-H: 40.2, AR-HA: 37.7, AR-A 22, and T: 46.4), making this system, in all configurations slightly less workload intensive than the average task that has been rated using the TLX. A mean score of 40.2 places the AR-H option near the top 30% of all scores, and 37.7 places AR-HA comfortably between the top 20 and 30%. 22 has AR-A

well within the top 10%, while 46.4 places T between 40% and 50%. Grier (2015) categorised the systems, offering an insight into how different types of systems compare, including robot operation systems. Compared to other robot operation systems, all four control options placed in the top 50%, with all three AR options in the top 25%.

6.4.4 Can Assigned Responsibility expand the capabilities of a robot beyond those currently automatable?

This thesis has argued that mixed control strategies are a better way of performing teleoperation tasks for two reasons. Firstly, direct control teleoperation can put significant workloads on human operators, workloads that can be reduced with the introduction of automation to teleoperation systems. Secondly, current automation techniques are not as a rule capable enough to provide reliable and accurate control alone in complex uncontrolled scenarios, a problem which can be fixed with the support of human operators. The questions above have tackled the first of these reasons, and this question seeks to tackle the second.

The experiments described above did not include scenarios that could not be accomplished with automation alone, as shown by the capability of the AR-A option to accomplish the task (if somewhat unreliably). The results did, however, show that an Assigned Responsibility system can allow a robot to be controlled successfully through a task in a variety of levels of autonomy, from full manual control through to full automation. As with other adjustable autonomy systems the possibility exists to operate under manual control, avoiding the issue of an unautomatable task altogether as long as the task is accomplishable by a human operator. From that point of view then, Assigned Responsibility can expand the capability of a robot beyond those automatable.

A more important reading of the research question is: Can Assigned Responsibility allow automation to operate where it would not have been able to before or better than it would have been able to before? The goal failure data suggests this might be the case. While under direct human control, a goal was failed only once out of 160 possible failures, whereas under

automated control goals were failed 27 times. Each of those goal failures could have been corrected by a human operator temporarily taking over from the automation, as they were all a result of the automation losing track of where it was, getting stuck, or failing to find the tap. While the experiments were designed to not allow human intervention during automated passages, a real world Assigned Responsibility system would allow such interventions. These might be triggered by the goal accomplishment tracker, if it is set up to detect failure conditions as well as goal accomplishment, or simply by a human operator that happens to notice the problems the automation was having. The participants of the experiment reported high frustration levels when the robot failed a task under automated control, with many commenting that they would rather take over than sit there and watch.

6.5 Summary

This chapter presented the evaluation of Assigned Responsibility, describing the usability experiment performed using the Assigned Responsibility implementation described in the previous chapter. This experiment sought to answer the research questions posed in this thesis, collecting information on the effectiveness and efficiency of, and workload caused by, this Assigned Responsibility system, comparing them to those of a traditional direct teleoperation system.

Twenty volunteers were recruited for this experiment, each performing a complex task four times, using a different control option each run. The data collected during this experiment provided insight into the thesis' research questions, showing that Assigned Responsibility is a sound way of teleoperating a robot, outperforming a direct control teleoperation system in several key areas, relieving operators of workload through the use of automation, while allowing them to accomplish a series of complex tasks in significantly less time, when operating the same robot.

Chapter 7

Conclusions

7.1 Overview

This thesis focused on the unexplored gap in the spectrum of possible strategies for mode switching teleoperated robots with adjustable autonomy. In situations where the environment is known and/or predictable, these control changes can be pre-planned, relieving robot operators of this additional task. This Assigned Responsibility strategy provides a clear division of labour between the automation and the human operator(s) before the job even begins, allowing for individual responsibilities to be known to the operator ahead of time, thus limiting confusion and allowing breaks to be planned, for example.

This strategy sacrifices some flexibility and reactivity in the face of unforeseen events over other adjustable autonomy strategies by pre-setting the levels of autonomy before a task begins. This however brings along some interesting advantages over other strategies in situations where this pre-planning can take place:

- Ideal operation: The system will always be in an ideal operating mode for a particular goal (assuming the plan was set up correctly). This removes any uncertainties the operator might have when deciding if or how to change the level of autonomy.
- Clear boundaries for the automation: By segmenting tasks into self-contained goals, it is possible to deploy targeted automation techniques

that do not need to be able to cope with the broader context of the task. This also has the advantage of easing the introduction of new techniques.

- Context for the operators: By clearly tracking the progress of the task, a human operator can always get the current context for a goal's accomplishment, facilitating the integration of newly arriving operators (e.g.: those returning from a break, or as a result of a swapping of operators).

- Clear accountability: The responsibilities for goal accomplishment being set out ahead of time means that in case of errors or misuse of the system, a clear chain of accountability is recorded. This information can be collected over time, and the suitability of a level of automation for a goal can be formally evaluated against others.

To evaluate this idea, this thesis proposed an assigned responsibility strategy, with an architecture supporting these pre-planned changes. A practical implementation of Assigned Responsibility was produced and evaluated through engineering tests and a usability study, demonstrating the viability of this approach as well as offering insight into its potential strengths and weaknesses.

The remainder of this chapter presents the conclusion of the thesis. Section 7.2 reviews the main argument of the thesis, with the major and minor contributions of this work summarised in Sections 7.3 and 7.4 respectively. Section 7.5 suggests future possibilities for this research as well as highlighting useful information gathered during this work that could be useful to others going forward. Finally, Section 7.6 provides closing remarks to the thesis.

7.2 Overview of the Thesis

Chapter 1 provides some background on the teleoperation of robots, first emphasising the difficulties of this task, then describing how these obstacles have been approached. The chapter discussed human-centred and automation-centred approaches to the remote control challenge,

focusing on how these approaches tackle this difficult task, as well as outlining their upsides and downsides. The chapter concluded that there were opportunities to improve this remote control by employing both approaches side-by-side.

Chapter 2 offers a deeper analysis of teleoperation, defining it and explaining its basics. The chapter then moves on to describing the scenarios where teleoperation is used, offering insight into the motivations for solving teleoperation issues. A look at how automation is used in teleoperation systems is given, leading to the idea of the spectrum of control modes, the realisation that automation in teleoperation is not an all or nothing approach, but rather a spectrum of possibilities ranging from no automation to full automation. A comparison of six teleoperation systems is provided, evaluating their purposes, operation and uses of automation. The chapter concludes that the current ideal solution to teleoperation is likely a flexible approach enabling the control mode to change during the course of a task.

Chapter 3 follows on by discussing the concept of adjustable autonomy, a flexible form of teleoperation that can change the amount of automation it makes use of during runtime. This is in opposition to traditional teleoperation interfaces that are designed in a much more static manner and only occupy one spot on the spectrum of control modes. The chapter explains the benefits of this approach over traditional teleoperation, and compares four adjustable autonomy implementations, focusing on their strategies to handle these changes in automation. This analysis finds that these implementations choose to change levels of automation in response to environmental changes rather than doing so in a pre-planned fashion, leaving a potential research area unexplored.

Chapter 4 proposes an architecture for Assigned Responsibility, a new form of adjustable autonomy teleoperation allowing changes in levels of autonomy to be pre-planned and to be executed automatically during runtime. To enable these automated changes to occur, the chapter also presents Goal Accomplishment Tracking, and architecture enabling a system to keep track of a task's progress to be tracked using sensors. The chapter ends by providing a design for an Assigned Responsibility system.

Chapter 5 describes the implementation of the design provided in the previous chapter. This implementation is split into two parts, hardware components, including the robot the system controls, as well as software components which make up the bulk of the work. The implementation is tested in several functional experiments, and found to be successful in achieving its goals of controlling the robot (manually and automatically), as well as providing the functions required by Assigned Responsibility.

Chapter 6 details the experimental work done to evaluate the idea of Assigned Responsibility. This is a usability experiment that measured the amount of workload human operators were put under as well as the quality of their control over the robot while they used a traditional teleoperation interface, as well as three different configurations of the Assigned Responsibility system. Results showed that Assigned Responsibility offers a significant decrease in operator workload, as well as an improvement in task completion time compared to traditional teleoperation, at the cost of a higher error rate due to limitations in the automation used.

7.3 Major Contributions

Assigned Responsibility for robot teleoperation.

This thesis introduced the idea of Assigned Responsibility, a new form of adjustable autonomy that enables the changes in levels of autonomy to be assigned before runtime. Assigned Responsibility is based around breaking down tasks into plans of goals and assigning responsibility for the accomplishment of each of these goals to a human operator, automation, or a mix of both. This pre-planning offers benefits over more traditional dynamic approaches, such as allowing human operators to know ahead of time when they are needed. Assigned Responsibility does not prescribe any particular methods for the accomplishment of tasks, instead providing a framework for the checking of the accomplishment of these tasks. As such it is suitable for the operation of all types of robots, as well as other semi-automated systems.

Goal Accomplishment Tracking.

This thesis also introduced a new method of Goal Accomplishment Tracking, for tracking the progress of a task through the use of sensor information. Goal Accomplishment Tracking relies on a plan of goals structure to operate. By being aware of the current goal, a system can deploy appropriate sensors and check them against readings known to indicate goal accomplishment. Once this accomplishment is detected, the plan can be updated, and the system can start checking the accomplishment of the new goal. This subdivision of tasks allows relatively simple sensors to be used to detect complex states. Checking on the goal state rather than the entity accomplishing the goal also allows the detection of goals accomplished by other agencies, saving time when a plan calls for actions that have already been executed.

An experimental evaluation of Assigned Responsibility teleoperation.

The thesis presented an evaluation of Assigned Responsibility done first through implementing the design in a real robot, then functionally testing it through a usability experiment. In this experiment, twenty subjects operated the robot through the use of both traditional teleoperation and a variety of Assigned Responsibility configurations. This evaluation recorded both the amount of workload the operators were under, as well as the quality of the control they exerted over the robot. This evaluation showed that Assigned Responsibility is a viable approach to robot teleoperation, and highlighted the advantages of human/automation cooperation over human only or automation only teleoperation.

7.4 Minor Contributions

An architecture for Assigned Responsibility.

A minor contribution of this thesis is an architecture for Assigned Responsibility based teleoperation. This architecture focuses on the proper management of the levels of autonomy of the teleoperation system, allowing changes in these levels to be setup ahead of time, ensuring the

system is automatically in the best possible level for each stage of the task. While the example used in this thesis is the teleoperation of a robot, this architecture could also easily be adapted to any system that could benefit from the inclusion of both human and automated operators.

An architecture for Goal Accomplishment Tracking.

Assigned Responsibility is not possible without accurate tracking of a task's progress. Goal Accomplishment Tracking is designed for this very purpose, enabling a system to test whether or not a goal is accomplished based on sensor tests. This architecture supports the management of the task's plan, the sensors required for the tests, and the tests themselves, coordinating these elements to achieve this tracking. This component could be used in other plan-based work systems.

A comparative analysis of 6 teleoperation interfaces.

Chapter 2 presented an analysis of six teleoperation systems. This analysis compared these systems on several factors, including their role, their control mode, and their tolerance to neglect. This analysis showed that manual control is still the dominant type of control, due to the need to avoid errors. However, it also showed that supervisory control modes are becoming more prevalent in situations where automation is more reliable, and when direct manual control is simply unfeasible.

A comparative analysis of 5 adjustable autonomy implementation.

This analysis applied the method for characterising a system's level of autonomy proposed by Parasuraman et al. (2000) to each level of autonomy of five adjustable autonomy implementations. This analysis highlighted the fact that the levels of autonomy used by different systems tend to be unique to those systems, showcasing wide the variety of possible levels available. This analysis also focused on the strategies employed by these systems to trigger changes in levels of autonomy. These systems were arranged along two axes describing their approaches to these changes: manual to automated, and dynamic to planned. A range of strategies were found, with none using planned automated changes.

7.5 Future Work and Recommendations

Further evaluation of Assigned Responsibility and adjustable autonomy.

The experiments shown in this thesis showed that Assigned Responsibility is a viable form of robot teleoperation, capable of accomplishing the tasks set out to operators while imposing less workload on these operators when compared to traditional teleoperation approaches. This study however can only hint at the situations where Assigned Responsibility should be deployed over other adjustable autonomy approaches. Are planned changes always easier on the human operators? Are automated changes a viable option when they could catch the human operators unaware? Are manual changes just another task human operators would rather avoid to have to do? Experiments comparing these strategies in changes in levels of autonomy are needed to properly answer these questions.

Recommendations for automated changes in autonomy.

It became apparent during the trials conducted for this research, that even when planned, automated changes in levels of autonomy could be an issue for human operators. During the trials, one goal was to line up the robot with a tap, a goal which was set to the H/A level of autonomy (human control with assistance from the automation). The automation assisted by highlighting the tap in the video feed to indicate to the human operator that the robot was detecting it. The next goal was to close the tap, set to full automation. When the tap was detected by the progress tracker, the goal would change, and the automation would take over.

This drastic change in level of autonomy, even if planned, caused a lot of frustration to the operators, as the goal might be considered accomplished before they were happy with the placement of the robot. Once the automation took over, the human operator was locked out of the control loop, adding to that frustration if the automation was struggling with closing of the tap due to a poor start position. This proved to be so frustrating, that some operators tried to hide the tap from the robot's

camera until they were satisfied with their position.

This is obviously an undesirable outcome when the goal is to reduce the operator's workload. An easy fix would be to make the operator one of the "sensors" for the goal accomplishment tracking, allowing them to report their satisfaction with the robot's state when ready. It seems then that some consideration must be paid to the triggers of automated changes in levels of autonomy (planned or not) to avoid inadvertently increasing operator workload.

On changes to the plan.

In case of errors, or unforeseen changes in the environment, it is likely that the plan as designed before the task started will no longer produce the desired results. Therefore, a strategy for handling divergences from this plan must be decided on. This problem was not addressed in the thesis directly, as the thesis dealt only with the core concepts of Assigned Responsibility, but potential solutions can be imagined and discussed here.

Plan-altering situations can be considered to fit either of two categories: i) smaller problems that require a correction before the plan can be resumed, and ii) major problems that will prevent the plan from ever completing. An example of the first would be the loss of a tool required for a future goal in the plan. Before the plan can be completed, the tool must either be found again, or another acquired. Once either has happened, the plan can resume from where it left off. A major issue would be the destruction of a piece of equipment that the plan required the robot to repair. The equipment can now no longer be repaired, rendering the plan unaccomplishable.

It is likely that the major issues would result in a simple cancellation of the plan rather than any kind of in-task re-planning, with a change in the mission objectives of the robot. More interestingly, the smaller problems require on-the-spot inclusion of additional steps (or the replacement of current steps), and thus the modification of the existing plan. Who makes these changes is an important question, as keeping operators aware of the plan is a requirement for Assigned Responsibility.

A low-tech solution could be to give full control of the robot to the human operator when such a problem arises, with the express goal of

returning the situation to an acceptable state. This defeats some of the benefits of Assigned Responsibility however, by imposing unexpected tasks on the human operator, and forcing a single control mode to be used. To make use of the full capabilities of AR, some amount of local re-planning must be done, either automatically or manually. This re-planning needs to add goals, and assign them a level of autonomy. Once the re-planning is done, the modifications must be made available to the operator for approval, ensuring all of the actors in the system are aware of their responsibilities before the task resumes. It is possible that for common problems a library of patches to the plan could exist, which could be rapidly deployed when these problems arise. This could somewhat reduce the issue of additional operator workload, as these patches would likely be familiar to operators that use the system often.

7.6 Concluding Remarks

While full automation of the robots used today in teleoperated tasks may be desirable, it is in many cases infeasible. In the immediate future, a mix of automation and manual control seems like the most promising avenue for useful robot control in most scenarios. Classic direct control teleoperation still reigns however, as it is hard to beat the reliability of human operators in most scenarios. If automation is still too untrustworthy to be charged with taking over control, the introduction of automation to teleoperation should then focus first and foremost on assisting the human operators doing this control work. Let automation take over the small simple tasks that still require time and effort from a human operator, let them spend that time and effort on more important tasks, like accomplishing their primary goals. As time passes and automation techniques become more refined and reliable, more tasks can be automated, slowly reducing the human operator's workload.

Assigned Responsibility was designed in part to help this process of gradual automation. The breakdown of the overall task into goals and subgoals sets delimitations that can be used to limit the influence of automation to the simple tasks without jeopardising the greater plan. This

also provides context for automation research, breaking down a complex task into a series of simpler tasks, that can be automated one by one instead of as a whole. While that is happening, the robot can keep on working, with the non automated parts of the plan handled by a human operator.

While this thesis was motivated by the specific problems facing robot control systems and their users, the concepts behind Assigned Responsibility are applicable to any system capable of adjustable autonomy. Such systems will be much more common in the near future, for example in agricultural equipment . How and when control should be passed from the human operator to the automation, and vice-versa, are important research questions that have to be answered sooner rather than later. Should those changes be reactive, planned, or a mixture of both? Human triggered, or automated? Assigned Responsibility is one strategy amongst others to manage these changes, making it either the answer, or a necessary consideration on the way to the answer.

This thesis has demonstrated that an Assigned Responsibility system can be used to successfully control a robot through a task using a variety of configurations. It is the author's hope that the research presented in this thesis will contribute to research in adjustable autonomy teleoperation, supporting further work in the mixing of automation and human-based control of remote robots, as well as provide insight into the wider questions governing human-automation interactions.

Bibliography

Adams, J. & Kaymaz-Keskinpala, H. (2004). Analysis of perceived workload when using a PDA for mobile robot teleoperation. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on* (Vol. 4, pp. 4128–4133).

Ambrose, R. O., Aldridge, H., Askew, R. S., Burridge, R. R., Bluethmann, W., Diftler, M., … Rehnmark, F. (2000). Robonaut: NASA's space humanoid. *Intelligent Systems and their Applications, IEEE*, *15*(4), 57–63.

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009, May). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, *57*(5), 469–483. doi:10.1016/j.robot.2008.10.024

Autor, D. H., Levy, F., & Murnane, R. J. (2001). *The skill content of recent technological change: An empirical exploration*. National Bureau of Economic Research.

Bajracharya, M., Maimone, M. W., & Helmick, D. (2008). Autonomy for mars rovers: Past, present, and future. *Computer*, *41*(12), 44–50.

Baker, M., Casey, R., Keyes, B., & Yanco, H. (2004). Improved interfaces for human-robot interaction in urban search and rescue. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on* (Vol. 3, pp. 2960–2965).

Ball, M. & Callaghan, V. (2012). Explorations of Autonomy: an investigation of adjustable autonomy in intelligent environments. In *Intelligent Environments (IE), 2012 8th International Conference on* (pp. 114–121). IEEE.

Bangor, A., Kortum, P., & Miller, J. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of usability studies*, *4*(3), 114–123.

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer vision and image understanding*, *110*(3), 346–359.

Bejczy, A., Kim, W., & Venema, S. (1990, May). The phantom robot: predictive displays for teleoperation with time delay. In *, 1990 IEEE International Conference on Robotics and Automation, 1990. Proceedings* (546–551 vol.1). doi:10.1109/ROBOT.1990.126037

Bekier, M., Molesworth, B. R., & Williamson, A. (2011, April). Defining the drivers for accepting decision making automation in air traffic management. *Ergonomics*, *54*(4), 347–356. doi:10.1080/00140139.2011.558635

Biesiadecki, J. J., Leger, P. C., & Maimone, M. W. (2007). Tradeoffs between directed and autonomous driving on the mars exploration rovers. *The International Journal of Robotics Research*, *26*(1), 91–104.

Bluethmann, W., Ambrose, R., Diftler, M., Askew, S., Huber, E., Goza, M., … Magruder, D. (2003). Robonaut: A robot designed to work with humans in space. *Autonomous Robots*, *14*(2), 179–197.

Bradshaw, J. M., Feltovich, P. J., Jung, H., Kulkarni, S., Taysom, W., & Uszok, A. (2004). Dimensions of adjustable autonomy and mixed-initiative interaction. In *Agents and Computational Autonomy* (pp. 17–39). Springer.

Bradshaw, J. M., Hoffman, R. R., Woods, D. D., & Johnson, M. (2013). The Seven Deadly Myths of "Autonomous Systems". *IEEE Intelligent Systems*, *28*(3), 54–61.

Brogårdh, T. (2007). Present and future robot control development—An industrial perspective. *Annual Reviews in Control*, *31*(1), 69–79. doi:10.1016/j.arcontrol.2007.01.002

Brooke, J. (1996). SUS-A quick and dirty usability scale. *Usability evaluation in industry*, *189*(194), 4–7.

Brown, C. (2012). Autonomous Vehicle Technology in Mining. *Engineering and Mining Journal*, *213*(1), 30.

Camarillo, D. B., Krummel, T. M., & Salisbury, J. K. (2004). Robotic technology in surgery: past, present, and future. *The American Journal of Surgery*, *188*(4), 2–15.

Carey, M. W., Kurz, E. M., Matte, J. D., Perrault, T. D., & Padir, T. (2012). Novel eod robot design with dexterous gripper and intuitive teleoperation. In *World Automation Congress (WAC), 2012* (pp. 1–6). IEEE.

Chen, J., Haas, E., & Barnes, M. (2007, November). Human Performance Issues and User Interface Design for Teleoperated Robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *37*(6), 1231–1245. doi:10.1109/TSMCC.2007.905819

Ciuti, G., Salerno, M., Lucarini, G., Valdastri, P., Arezzo, A., Menciassi, A., … Dario, P. (2012, April). A Comparative Evaluation of Control Interfaces for a Robotic-Aided Endoscopic Capsule Platform. *IEEE Transactions on Robotics*, *28*(2), 534–538. doi:10.1109/TRO.2011.2177173

Collet, A., Berenson, D., Srinivasa, S. S., & Ferguson, D. (2009). Object recognition and full pose registration from a single image for robotic manipulation. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* (pp. 48–55). IEEE.

Côté, N., Bouzid, M., & Mouaddib, A.-I. (2011). Integrating the human recommendations in the decision process of autonomous agents: A goal biased markov decision process. In *Proceedings of the AAAI 2011 Fall Symposium Robot-Human Teamwork in Dynamic Adverse Environmen*.

Côté, N., Canu, A., Bouzid, M., & Mouaddib, A.-I. (2012). Humans-robots sliding collaboration control in complex environments with adjustable autonomy. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on* (Vol. 2, pp. 146–153). IEEE.

DARPA. (2015, June). Home | DRC Finals.

Das, H., Sheridan, T., & Slotine, J.-J. (1989, November). Kinematic control and visual display of redundant teleoperators. In *, IEEE*

International Conference on Systems, Man and Cybernetics, 1989. Conference Proceedings (1072–1077 vol.3). doi:10.1109/ICSMC.1989.71462

Dastani, M., Riemsdijk, M. B. V., & Meyer, J.-j. C. (2006). Goal types in agent programming. In *In Proceedings of the 17th European Conference on Artificial Intelligence (ECAI'06* (pp. 220–224).

de Jong, I. (2016). Pyro - Python Remote Objects - 4.41 — Pyro 4.41 documentation.

Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., … Livny, M. (2004). Pegasus: Mapping scientific workflows onto the grid. In *Grid Computing* (pp. 11–20).

Diftler, M. A., Ahlstrom, T. D., Ambrose, R. O., Radford, N. A., Joyce, C. A., De La Pena, N., … Noblitt, A. L. (2012). Robonaut 2—initial activities on-board the ISS. In *Aerospace Conference, 2012 IEEE* (pp. 1–12). IEEE.

Diftler, M. A., Culbert, C. J., Ambrose, R. O., Platt Jr, R., & Bluethmann, W. J. (2003). Evolution of the NASA/DARPA robonaut control system. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on* (Vol. 2, pp. 2543–2548).

Diftler, M., Mehling, J., Abdallah, M., Radford, N., Bridgwater, L., Sanders, A., …, Permenter, F., et al. (2010). Robonaut 2 the first humanoid robot in space. In *IEEE Int'l Conference on Robotics and Automation (Submitted)*.

Dorais, G., Bonasso, R. P., Kortenkamp, D., Pell, B., & Schreckenghost, D. (1999). Adjustable autonomy for human-centered autonomous systems. In *Working notes of the Sixteenth International Joint Conference on Artificial Intelligence Workshop on Adjustable Autonomy Systems* (pp. 16–35).

Drury, J. L., Scholtz, J., & Yanco, H. A. (2004, April). Applying CSCW and HCI techniques to human-robot interaction. In *CHI 2004 Workshop on Shaping Human-Robot Interaction* (pp. 13–16). Vienna.

Drury, J. L., Scholtz, J., & Yanco, H. A. (2006). *Applying CSCW and HCI techniques to human-robot interaction*. DTIC Document.

Durrant-Whyte, H. & Bailey, T. (2006). Simultaneous localization and mapping: part I. *Robotics & Automation Magazine, IEEE*, *13*(2), 99–110.

Falcone, R. & Castelfranchi, C. (1999). Levels of Delegation and Levels of Adoption as the basis for Adjustable Autonomy. In *Congress of the Italian Association for Artificial Intelligence* (pp. 273–284). Springer.

Fong, T. & Thorpe, C. (2001). Vehicle teleoperation interfaces. *Autonomous robots*, *11*(1), 9–18.

Fong, T., Thorpe, C., & Baur, C. (2001). Advanced interfaces for vehicle teleoperation: Collaborative control, sensor fusion displays, and remote driving tools. *Autonomous Robots*, *11*(1), 77–85.

Foundation, P. S. (n.d.). Welcome to Python.org.

Gettys, J. & Nichols, K. (2012, January). Bufferbloat: dark buffers in the internet. *Commun. ACM*, *55*(1), 57–65. doi:10.1145/2063176.2063196

Ghallab, M., Nau, D., & Traverso, P. (2004, May). *Automated Planning: Theory & Practice*. Elsevier.

Gingras, D., Allard, P., Lamarche, T., Rocheleau, S. G., Gemme, S., Deschênes-Villeneuve, L., & Martin, E. (2014). Lunar rover remote driving using monocameras under multi-second latency and low-bandwidth: Field tests and lessons learned. In *Submitted to International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS),(Montreal, Canada)*.

Glass, B. & Briggs, G. (2003). Evaluation of human vs. teleoperated robotic performance in field geology tasks at a Mars analog site.

Goodrich, M., Olsen, D., Crandall, J., & Palmer, T. (2001). Experiments in adjustable autonomy. In *Proceedings of IJCAI Workshop on Autonomy, Delegation and Control: Interacting with Intelligent Agents* (pp. 1624–1629).

Goodrich, M. A. & Schultz, A. C. (2007). Human-Robot Interaction: A Survey. *Foundations and Trends in Human-Computer Interaction*, *1*(3), 203–275. doi:10.1561/1100000005

Goza, S. M., Ambrose, R. O., Diftler, M. A., & Spain, I. M. (2004). Telepresence control of the NASA/DARPA robonaut on a mobility

platform. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 623–629).

Greicius, T. (2015, January). Curiosity Overview. Text.

Grier, R. A. (2015). How High is High? A Meta-Analysis of NASA-TLX Global Workload Scores. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 59, pp. 1727–1731). SAGE Publications.

Grotzinger, J. P., Crisp, J., Vasavada, A. R., Anderson, R. C., Baker, C. J., Barry, R., … Wiens, R. C. (2012, July). Mars Science Laboratory Mission and Science Investigation. *Space Science Reviews*, *170*(1-4), 5–56. doi:10.1007/s11214-012-9892-2

Hainsworth, D. (2001). Teleoperation user interfaces for mining robotics. *Autonomous Robots*, *11*(1), 19–28.

Hannaford, B., Wood, L., McAffee, D., & Zak, H. (1991, May). Performance evaluation of a six-axis generalized force-reflecting teleoperator. *IEEE Transactions on Systems, Man and Cybernetics*, *21*(3), 620–633. doi:10.1109/21.97455

Hart, S. G. & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Advances in psychology*, *52*, 139–183.

Hokayem, P. F. & Spong, M. W. (2006). Bilateral teleoperation: An historical survey. *Automatica*, *42*(12), 2035–2057.

Huang, H.-M., Pavek, K., Novak, B., Albus, J., & Messin, E. (2005). A framework for autonomy levels for unmanned systems (ALFUS). *Proceedings of AUVSI Unmanned Systems 2005*.

Ip, W.-H., Yan, J., Li, C.-L., & Ouyang, Z.-Y. (2014, December). Preface: The Chang'e-3 lander and rover mission to the Moon. *Research in Astronomy and Astrophysics*, *14*(12), 1511–1513. doi:10.1088/1674-4527/14/12/001

Isaac, A. & Sammut, C. (2003). Goal-directed learning to fly. In *ICML* (pp. 258–265).

Jaju, A., Banerji, A., & Pal, P. (2013, October). Development and evaluation of a telepresence interface for teleoperation of a robot manipulator. In

2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI) (pp. 90–95). doi:10.1109/URAI.2013.6677481

Kaltenbrunner, M. & Bencina, R. (2007). reacTIVision: a computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st international conference on Tangible and embedded interaction* (pp. 69–74). ACM.

Kephart, J. O. & Chess, D. M. (2003). The vision of autonomic computing. *Computer*, *36*(1), 41–50.

Kohlbrecher, S., Romay, A., Stumpf, A., Gupta, A., Von Stryk, O., Bacim, F., … Conner, D. C. (2015). Human-robot Teaming for Rescue Missions: Team ViGIR's Approach to the 2013 DARPA Robotics Challenge Trials. *Journal of Field Robotics*, *32*(3), 352–377.

Kortenkamp, D., Keirn-Schreckenghost, D., & Bonasso, R. P. (2000). Adjustable control autonomy for manned space flight. In *Aerospace Conference Proceedings, 2000 IEEE* (Vol. 7, pp. 629–640). IEEE.

Kumar, V. & Mason, M. (2011). Are we even in the game? In *Berlin Summit on Robotics 2011: Conference Report* (pp. 16–24). Berlin.

Lawrence, M. A. (2013). *Package 'ez'*. Sep.

Lee, J. D. (2008). Review of a pivotal Human Factors article:"Humans and automation: use, misuse, disuse, abuse". *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *50*(3), 404–410.

Lee, J. D. & See, K. A. (2004). Trust in automation: Designing for appropriate reliance. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *46*(1), 50–80.

Lee, K., Paton, N. W., Sakellariou, R., Deelman, E., Fernandes, A. A., & Mehta, G. (2009). Adaptive workflow processing and execution in pegasus. *Concurrency and Computation: Practice and Experience*, *21*(16), 1965–1981.

Lepetit, V. & Fua, P. (2005). Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and trends in computer graphics and vision*, *1*(CVLAB-ARTICLE-2005-002), 1–89.

Leymann, F., Roller, D., & Thatte, S. (n.d.). Goals of the BPEL4ws Specification.

Liu, Y. & Nejat, G. (2013, November). Robotic Urban Search and Rescue: A Survey from the Control Perspective. *Journal of Intelligent & Robotic Systems*, *72*(2), 147–165. doi:10.1007/s10846-013-9822-x

Mann, G., Small, N., Lee, K., Clarke, J., & Sheh, R. (2015, September). Field-Testing Astronaut Assistance Robots in Australian Outback [From the Field]. *IEEE Robotics Automation Magazine*, *22*(3), 188–191. doi:10.1109/MRA.2015.2452200

Mann, G. A. (2008). Quantitative evaluation of human-robot options for maintenance tasks during analogue surface operations. In *Proceedings of the 8th Australian Mars Exploration Conference* (pp. 26–34).

Mann, G. A. & Small, N. (2012). Opportunities for enhanced robot control along the adjustable autonomy scale. In *Human System Interactions (HSI), 2012 5th International Conference on* (pp. 35–42). IEEE.

Marohn, C. M. R. & Hanly, C. E. J. (2004). Twenty-first century surgery using twenty-first century technology: surgical robotics. *Current Surgery*, *61*(5), 466–473.

mars.nasa.gov. (n.d.). Curiosity Self-Portrait at 'Okoruso' Drill Hole, Mars - Mars Science Laboratory.

Martinez-Tenor, A. & Fernandez-Madrigal, J.-A. (2015, June). Smoothly adjustable autonomy for the low-level remote control of mobile robots that is independent of the navigation algorithm. In *2015 23th Mediterranean Conference on Control and Automation (MED)* (pp. 1071–1078). doi:10.1109/MED.2015.7158899

Mast, M., Materna, Z., Španěl, M., Weisshardt, F., Arbeiter, G., Burmester, M., … Graf, B. (2015, April). Semi-Autonomous Domestic Service Robots: Evaluation of a User Interface for Remote Manipulation and Navigation With Focus on Effects of Stereoscopic Display. *International Journal of Social Robotics*, *7*(2), 183–202. doi:10.1007/s12369-014-0266-7

Matsuno, F., Sato, N., Kon, K., Igarashi, H., Kimura, T., & Murphy, R. (2014). Utilization of robot systems in disaster sites of the great eastern japan earthquake. In *Field and Service Robotics* (pp. 1–17). Springer.

Melvin, W. S., Needleman, B. J., Krause, K. R., Schneider, C., Wolf, R. K., Michler, R. E., & Ellison, E. C. (2002). Computer-enhanced robotic

telesurgery. *Surgical Endoscopy And Other Interventional Techniques*, *16*(12), 1790–1792.

Miller, C. A. & Parasuraman, R. (2007). Designing for flexible interaction between humans and automation: Delegation interfaces for supervisory control. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *49*(1), 57–75.

Miller, C. & Parasuraman, R. (2003). Beyond levels of automation: An architecture for more flexible human-automation collaboration. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 47, pp. 182–186).

Minguez, J. & Montano, L. (2005). Sensor-based robot motion generation in unknown, dynamic and troublesome scenarios. *Robotics and Autonomous Systems*, *52*(4), 290–311.

Murphy, R. R. (2004, March). Rescue Robotics for Homeland Security. *Commun. ACM*, *47*(3), 66–68. doi:10.1145/971617.971648

Murphy, R. R. & Burke, J. L. (2005, September). Up from the Rubble: Lessons Learned about HRI from Search and Rescue. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *49*(3), 437–441. doi:10.1177/154193120504900347

Muszynski, S., Stuckler, J., & Behnke, S. (2012). Adjustable autonomy for mobile teleoperation of personal service robots. In *RO-MAN, 2012 IEEE* (pp. 933–940). IEEE.

Nagatani, K., Kiribayashi, S., Okada, Y., Otake, K., Yoshida, K., Tadokoro, S., … Kawatsuma, S. (2013, January). Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots. *Journal of Field Robotics*, *30*(1), 44–63. doi:10.1002/rob.21439

Nagatani, K., Kiribayashi, S., Okada, Y., Tadokoro, S., Nishimura, T., Yoshida, T., … Hada, Y. (2011). Redesign of rescue mobile robot Quince. In *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on* (pp. 13–18). IEEE.

Nielsen, C. & Goodrich, M. (2006). Comparing the usefulness of video and map information in navigation tasks. In *Proceedings of the 1st ACM*

SIGCHI/SIGART conference on Human-robot interaction (pp. 95–101).

Nielsen, C., Goodrich, M., & Ricks, R. (2007, October). Ecological Interfaces for Improving Mobile Robot Teleoperation. *IEEE Transactions on Robotics*, *23*, 927–941. doi:10.1109/TRO.2007.907479

Nielsen, J. (1993, September). *Usability Engineering* (1 edition). Boston: Morgan Kaufmann.

Olivares, R., Zhou, C., Bodenheimer, B., & Adams, J. A. (2003). Interface evaluation for mobile robot teleoperation. In *Proceedings of the ACM Southeast Conference (ACMSE03)* (Vol. 112, p. 118).

Olsen, D. R. & Goodrich, M. A. (2003). Metrics for evaluating human-robot interactions. In *Proceedings of PERMIS* (Vol. 2003, p. 5).

Ono, M., Fuchs, T., Steffy, A., Maimone, M., & Yen, J. (2015, March). Risk-aware planetary rover operation: Autonomous terrain classification and path planning. In *2015 IEEE Aerospace Conference* (pp. 1–10). doi:10.1109/AERO.2015.7119022

NASA TLX Paper and Pencil Version Instruction Manual. (n.d.). Human Performance Rsearch Group, NASA Ames Research Center.

Palep, J. H. (2009). Robotic assisted minimally invasive surgery. *Journal of Minimal Access Surgery*, *5*(1), 1–7. doi:10.4103/0972-9941.51313

Parasuraman, R. & Riley, V. (1997). Humans and automation: Use, misuse, disuse, abuse. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *39*(2), 230–253.

Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *30*(3), 286–297.

Parasuraman, R. & Miller, C. (2006). 18. Delegation Interfaces for Human Supervision of Multiple Unmanned Vehicles: Theory, Experiments, and Practical Applications. In *Advances in Human Performance and Cognitive Engineering Research* (Vol. 7, pp. 251–266). Elsevier.

Perzanowski, D., Schultz, A. C., Adams, W., & Marsh, E. (1999). Goal tracking in a natural language interface: Towards achieving adjustable

autonomy. In *Computational Intelligence in Robotics and Automation, 1999. CIRA'99. Proceedings. 1999 IEEE International Symposium on* (pp. 208–213).

Pratt, K. S. & Murphy, R. R. (2012). Protection from human error: Guarded motion methodologies for mobile robots. *IEEE Robotics & Automation Magazine*, *19*(4), 36–47.

Randell, B. (1975). System structure for software fault tolerance. *Software Engineering, IEEE Transactions on*, (2), 220–232.

Rehnmark, F., Bluethmann, W., Mehling, J., Ambrose, R., Diftler, M., Chu, M., & Necessary, R. (2005). Robonaut: the 'Short List' of technology hurdles. *Computer*, *38*(1), 28–37.

Rosen, J., Hannaford, B., MacFarlane, M., & Sinanan, M. (1999, October). Force controlled and teleoperated endoscopic grasper for minimally invasive surgery-experimental performance evaluation. *IEEE Transactions on Biomedical Engineering*, *46*(10), 1212–1221. doi:10.1109/10.790498

Sacerdoti, E. (1977). *A Structure For Plans and Behavior*. Elsevier Publishing Company.

Scholtz, J., Young, J., Drury, J., & Yanco, H. (2004, April). Evaluation of human-robot interaction awareness in search and rescue. (Vol. 3, 2327–2332 Vol.3). doi:10.1109/ROBOT.2004.1307409

Scholtz, J., Theofanos, M., & Antonishek, B. (2006). Development of a Test Bed for Evaluating Human-robot Performance for Explosive Ordnance Disposal Robots. (pp. 10–17). HRI '06. New York, NY, USA: ACM. doi:10.1145/1121241.1121246

Sellner, B., Heger, F. W., Hiatt, L. M., Simmons, R., & Singh, S. (2006). Coordinated multiagent teams and sliding autonomy for large-scale assembly. *Proceedings of the IEEE*, *94*(7), 1425–1444.

Sheridan, T. B. (1992). *Telerobotics, automation, and human supervisory control*. MIT Press.

Sheridan, T. B. & Verplanck, W. L. (1978). *Human and computer control of undersea teleoperators*. MIT Man-Machine Systems Laboratory. Cambridge, Mass.

Shneiderman, B., Plaisant, C., Cohen, M., & Jacobs, S. (2009, March). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (5th ed.). Addison Wesley.

Sierhuis, M., Bradshaw, J. M., Acquisti, A., Van Hoof, R., Jeffers, R., & Uszok, A. (2003). Human-agent teamwork and adjustable autonomy in practice. In *Proceedings of the seventh international symposium on artificial intelligence, robotics and automation in space (I-SAIRAS).*

Smith, R., Day, A., Rockall, T., Ballard, K., Bailey, M., & Jourdan, I. (2012, January). Advanced stereoscopic projection technology significantly improves novice performance of minimally invasive surgical skills. *Surgical Endoscopy*, *26*(6), 1522–1527. doi:10.1007/s00464-011-2080-8

Sonntag, M., Karastoyanova, D., & Deelman, E. (2010). Bridging the Gap between Business and Scientific Workflows: Humans in the Loop of Scientific Workflows. In *E-Science (e-Science), 2010 IEEE Sixth International Conference on* (pp. 206–213).

Stentz, A., Herman, H., Kelly, A., Meyhofer, E., Haynes, G. C., Stager, D., …, Dellin, C., et al. (2015). Chimp, the cmu highly intelligent mobile platform. *Journal of Field Robotics*, *32*(2), 209–228.

Sugimoto, M., Kagotani, G., Nii, H., Shiroma, N., Matsuno, F., & Inami, M. (2005, January). Time Follower's Vision: a teleoperation interface with past images. *IEEE Computer Graphics and Applications*, *25*(1), 54–63. doi:10.1109/MCG.2005.23

Sung, G. T. & Gill, I. S. (2001, December). Robotic laparoscopic surgery: a comparison of the da Vinci and Zeus systems. *Urology*, *58*(6), 893–898. doi:10.1016/S0090-4295(01)01423-6

Tayebi, A. & McGilvray, S. (2004, December). Attitude stabilization of a four-rotor aerial robot. In *43rd IEEE Conference on Decision and Control, 2004. CDC* (Vol. 2, 1216–1221 Vol.2). doi:10.1109/CDC.2004.1430207

Thrun, S. (2000). Probabilistic algorithms in robotics. *AI Magazine*, *21*(4), 93.

Venkatraman, V. (2015, July). The space roboticist. *Science*, *349*(6245), 338–338. doi:10.1126/science.349.6245.338

Wall, J. & Marescaux, J. (2013). History of telesurgery. In *Telemicrosurgery* (pp. 15–18). Springer.

Washington, R., Golden, K., Bresina, J., Smith, D. E., Anderson, C., & Smith, T. (1999). Autonomous rovers for Mars exploration. In *Aerospace Conference, 1999. Proceedings. 1999 IEEE* (Vol. 1, pp. 237–251). IEEE.

Wright, J., Hartman, F., Maxwell, S., Cooper, B., & Yen, J. (2013, June). Updates to the rover driving tools for Curiosity. In *2013 8th International Conference on System of Systems Engineering (SoSE)* (pp. 147–152). doi:10.1109/SYSoSE.2013.6575258

Yi, S.-J., McGill, S. G., Vadakedathu, L., He, Q., Ha, I., Han, J., …, Hong, D., et al. (2015). Team THOR's Entry in the DARPA Robotics Challenge Trials 2013. *Journal of Field Robotics*, *32*(3), 315–335.

Zieba, S., Polet, P., & Vanderhaegen, F. (2011). Using adjustable autonomy and human–machine cooperation to make a human–machine system resilient–Application to a ground robotic system. *Information Sciences*, *181*(3), 379–397.

Zieba, S., Polet, P., Vanderhaegen, F., & Debernard, S. (2008, December). Resilience of a human-robot system using adjustable autonomy and human-robot collaborative control. *International Journal of Adaptive and Innovative Systems*, *1*(1), 13–29. doi:10.1504/IJAIS.2009.022

Zieba, S., Polet, P., Vanderhaegen, F., & Debernard, S. (2010). Principles of adjustable autonomy: a framework for resilient human–machine cooperation. *Cognition, Technology & Work*, *12*(3), 193–203.

APPENDIX A. XML SCHEMA

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="plan—tree">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="goal" type="goalDef">
                                </xs:element>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <xs:complexType name="goalDef">
                <xs:sequence>
                        <xs:element name="name" type="xs:string" />
                        <xs:element name="assignment" minOccurs="0">
                                <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                                <xs:enumeration value="H" />
                                                <xs:enumeration value="H/A" />
                                                <xs:enumeration value="A/H" />
                                                <xs:enumeration value="A" />
                                        </xs:restriction>
                                </xs:simpleType>
                        </xs:element>
                        <xs:element name="status">
                                <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                                <xs:enumeration value="Accomplished" />
                                                <xs:enumeration value="Not Started" />
                                                <xs:enumeration value="Started" />
                                        </xs:restriction>
                                </xs:simpleType>
                        </xs:element>
                        <xs:element name="subgoal" maxOccurs="unbounded" minOccurs="0"
                             type="goalDef" />
        </xs:sequence>
                <xs:attribute  name="goal—type" use="required">
                                <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                                <xs:enumeration value="Parent"/>
                                                <xs:enumeration value="Achieve"/>
                                                <xs:enumeration value="Maintain"/>
                                                <xs:enumeration value="Opportunistic"/>
                                        </xs:restriction>
                                </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="uid">
        </xs:complexType>
</xs:schema>
```

Figure A.1: XML Schema Definition for the System's Plans

APPENDIX C. USABILITY EXPERIMENT: EXPLANATIONS TO PARTICIPANTS

Note: This section is read while observing the experimental setup. Each mode is explained when the relevant trial is being described.

The trial takes place in this setup which is divided in four rooms as well as a central corridor. Each trial consists of twelve goals that have to be completed in order. These goals can be described as sets of three goals that have to be repeated four times; once in each room.

The three goals are:

1. Go to Sector:

   Accomplishing this goal is a navigation task. The robot needs to travel from its current location to the specified sector (1, 2, 3, or 4).

2. Find Valve:

   This goal is accomplished when the arm-mounted camera is facing the valve. The robot must be used to search the sector until this valve is found.

3. Close Valve:

   The goal is accomplished when the valve is closed. The robot's arm must be used to reach for and turn the valve in order to close it.

You will need to accomplish these goals using a different control method in each trial. You will undertake each trial once in a random order.

Below are detailed instructions on accomplishing the goals using each option.

a. "T" Trial: Teleoperated Control

In teleoperated control, you have manual control over the robot's actions, and must confirm the results of these actions themselves as there is no automated progress tracking. Use the controller to guide the robot through the goals in order. A pen and paper will be provided to you if you wish to keep track of your progress manually.

You can use both the video and map screens to assist you.

b. "AR/H" Trial: Human Only Assigned Responsibility

In this trial, you have manual control over the robot's actions, however your progress through the task is being tracked automatically. Use the controller to guide the robot through the goals in order.

Your progress will be shown in the plan screen, and the current goal will be listed at the bottom of each screen. You can use both the video and map screens to assist you.

c. "AR/HA" Trial: Human and Automation Assigned Responsibility

During this trial, each goal will be set to a particular mode: a collaboration of human and automation. The level of involvement you will have in each goal will vary on the mode.

The mode each gaol is set to is:

1. Go to Sector: The goal is set to A/H Mode (Automated with human supervision). You will be asked to switch to the map screen and input waypoints on the map to guide the robot to the desired location. Once the you submit the waypoints, the automation executes the required moves. If the automation does not manage to accomplish the move, you will be asked to enter new waypoints. Another failure requires the you to take manual control of the robot and finish the move.

2. Find Valve: The goal is set to H/A mode (Human control, Automation assisting). You must use the controller to guide the robot's movements, the automation will provide an overlay highlighting the valve's location in the camera feed if detected.

3. Close Valve: The goal is set to A mode (Full automation). You do not have control over any of the robot's actions. In case of repeated failure from the automation, you will be asked to take control and finish the task.

APPENDIX D. NASA TLX INSTRUCTIONS AND FORMS

These instructions are provided in the "NASA TLX Paper and Pencil Version Instruction Manual" (n.d.), and are used as is.

Table D.1: Rating Scale Descriptions

Title	Endpoints	Descriptions
MENTAL DEMAND	Low/High	How much mental and perceptual activity was required (e.g. thinking, deciding, calculating, remembering, looking, searching, etc.) Was the task easy or demanding, simple or complex, exacting or forgiving?
PHYSICAL DEMAND	Low/High	How much physical activity was required (e.g. pushing, pulling, turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?
TEMPORAL DEMAND	Low/High	How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?
PERFORMANCE	Good/Poor	How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?
EFFORT	Low/High	How hard did you have to work (mentally and physically) to accomplish your level of performance?
FRUSTRATION LEVEL	Low/High	How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed, and complacent did you feel during the task?

a. SUBJECT INSTRUCTIONS: RATING SCALES

We are not only interested in assessing your performance but also the experiences you had during the different task conditions. Right now we are going to describe the technique that will be used to examine your experiences. In the most general sense we are examining the "Workload" you experienced. Workload is a difficult concept to define precisely, but a simple one to understand generally. The factors that influence your experience of workload may come from the task itself, your feelings about your own performance, how much effort you put in, or the stress and frustration you felt. The workload contributed by different task elements

may change as you get more familiar with a task, perform easier or harder versions of it, or move from one task to another. Physical components of workload are relatively easy to conceptualise and evaluate. However, the mental components of workload may be more difficult to measure.

Since workload is something that is experienced individually by each person, there are no effective "rulers" that can be used to estimate the workload of different activities. One way to find out about workload is to ask people to describe the feelings they experienced. Because workload may be caused by many different factors, we would like you to evaluate several of them individually rather than lumping them into a single global evaluation of overall workload. This set of six rating scales was developed for you to use in evaluating your experiences during different tasks. Please read the descriptions of the scales carefully. If you have a question about any of the scales in the table, please ask me about it. It is extremely important that they be clear to you. You may keep the descriptions with you for reference during the experiment.

After performing each of the tasks, you will be given a sheet of rating scales. You will evaluate the task by putting an "X" on each of the six scales at the point which matches your experience. Each line has two endpoint descriptors that describe the scale. Note that "own performance" goes from "good" on the left to "bad" on the right. This order has been confusing for some people. Please consider your responses carefully in distinguishing among the different task conditions. Consider each scale individually. Your ratings will play an important role in the evaluation being conducted, thus, your active participation is essential to the success of this experiment and is greatly appreciated by all of us.

b. SUBJECT INSTRUCTIONS: SOURCES OF WORKLOAD EVALUATION

Throughout this experiment the rating scales are used to assess your experiences in the different task conditions. Scales of this sort are extremely useful, but their utility suffers from the tendency people have to interpret them in individual ways. For example, some people feel that

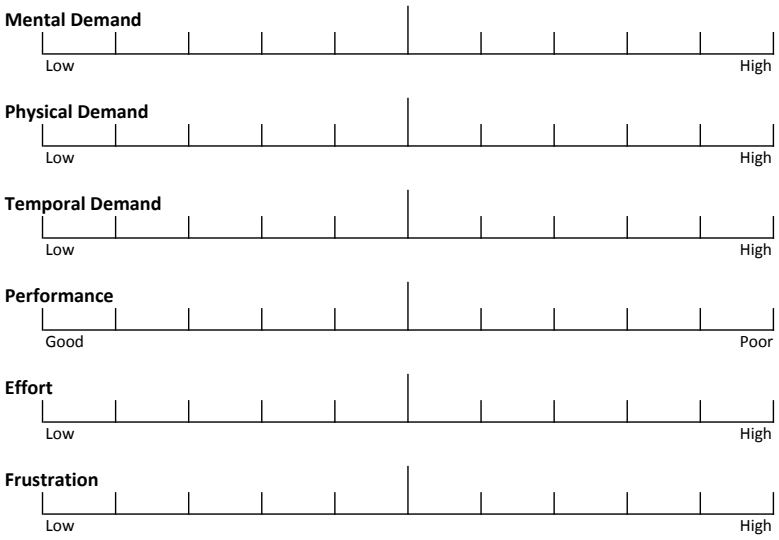
mental or temporal demands are the essential aspects of workload regardless of the effort they expended on a given task or the level of performance they achieved. Others feel that if they performed well the workload must have been low and if they performed badly it must have been high. Yet others feel that effort or feelings of frustration are the most important factors in workload; and so on. The results of previous studies have already found every conceivable pattern of values. In addition, the factors that create levels of workload differ depending on the task. For example, some tasks might be difficult because they must be completed very quickly. Others may seem easy or hard because of the intensity of mental or physical effort required. Yet others feel difficult because they cannot be performed well, no matter how much effort is expended.

The evaluation you are about to perform is a technique that has been developed by NASA to assess the relative importance of six factors in determining how much workload you experienced. The procedure is simple: You will be presented with a series of pairs of rating scale titles (for example, Effort vs. Mental Demands) and asked to choose which of the items was more important to your experience of workload in the task(s) that you just performed. Each pair of scale titles will appear on a separate card.

Place a tick next to the Scale Title that represents the more important contributor to workload for the specific task(s) you performed in this experiment. After you have finished the entire series we will be able to use the pattern of your choices to create a weighted combination of the ratings from that task into a summary workload score. Please consider your choices carefully and make them consistent with how you used the rating scales during the particular task you were asked to evaluate. Don't think that there is any correct pattern: we are only interested in your opinions.

If you have any questions, please ask them now. Otherwise start whenever you are ready. Thank you for your participation.

Task: AR-H **Name:** Participant Name



Sources of Workload **Name:** Participant Name

Effort or **Peformance**	**Temporal Demand** or **Frustration**	**Frustration** or **Effort**
Temporal Demand or **Effort**	**Physical Demand** or **Frustration**	**Peformance** or **Temporal Demand**
Peformance or **Frustration**	**Physical Demand** or **Temporal Demand**	**Mental Demand** or **Physical Demand**
Physical Demand or **Peformance**	**Temporal Demand** or **Mental Demand**	**Frustration** or **Mental Demand**
Peformance or **Mental Demand**	**Mental Demand** or **Effort**	**Effort** or **Physical Demand**

APPENDIX E. SYSTEM USABILITY SCALE INSTRUCTIONS AND FORMS

This questionnaire will allow us to record your impressions of the interface you used to control the robot. You will be presented with a series of statements and asked to respond by indicating on a scale how you feel about the interface. Record your immediate response to each item, rather than thinking about each of them for a long time. The scales range from 1 (Strongly disagree) to 5 (Strongly agree). Place a tick on each scale that represents how you feel about each statement with regards to the interface you just used.

All items should be checked. If you feel that you cannot respond to a particular item, you should mark the centre point of the scale.

System Usability Scale

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5